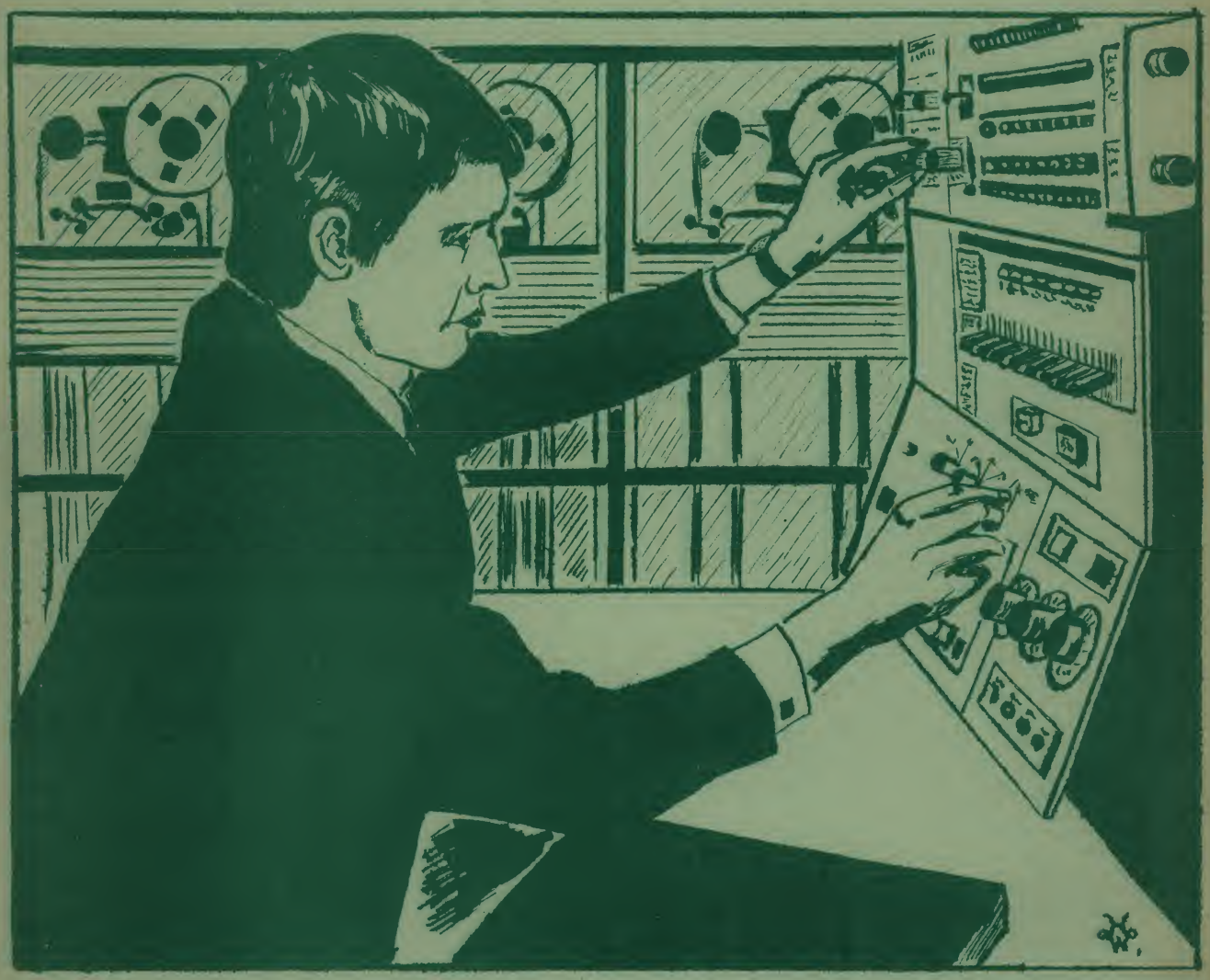
COMPUTER PROGRAMMEUR



Les 4

SUBPROGRAMMA'S, MODIFICATIE VAN OPDRACHTEN

Ir. P.A. Tas

Een publikatie van de Stichting Het Nederlands Studiecentrum voor Informatica, Amsterdam.

Ir. P.A. Tas

1. INLEIDING

Een van de zeer belangrijke eigenschappen van de "stored program" computer is wel dat de rekenmachine in staat is te rekenen met het adresgedeelte van instrukties, waardoor de dynamische uitvoering van het programma kan worden beïnvloed.

Het veranderen van het adresgedeelte kan principiëel op 2 verschillende wijzen uitgevoerd worden.
- *Het adresgedeelte van een instruktie wordt in het geheugen veranderd.*
- *Het adresgedeelte van een instruktie wordt in het besturingsorgaan veranderd en de instruktie blijft onveranderd in het geheugen staan.*

Het eerste vereist geen speciale opdracht en wordt " voor de voet schrijven" genoemd.
Bij de tweede manier wordt gebruik gemaakt van een speciale groep opdrachten : de modificatieopdrachten.

2. VOOR DE VOET SCHRIJVEN

Aangezien instrukties in een woord worden opgeborgen en er aan het woord niet te zien is of de inhoud een getal of een instruktie is, moet het mogelijk zijn om normaal te rekenen met instrukties, zoals dat ook gedaan wordt met getallen.
Wel zullen bepaalde woorden in het geheugen, waarin zo'n instruktie staat, veranderd worden waardoor er een verschil ontstaat ten opzichte van het geschreven programma.
Dit kan het beste duidelijk gemaakt worden aan de hand van een voorbeeld.

Voorbeeld :

Stel er moet een hoeveelheid informatie in het geheugen worden verplaatst.
We zullen aannemen dat dit 100 woorden betreft, die in het geheugen staan, te beginnen bij adres met label LYSTI.
Deze 100 woorden moeten worden verplaatst naar een stuk geheugen met beginadres LYST2.

```
        HPA *                  | 0 ——→ teller
        BPA   TELLER           |
INSTR1  HPA   LYST1   }        |
INSTR2  BPA   LYST2   } verplaats een woord
        HPA   INSTR1           | verhoog het adresgedeelte van de
        OPA * 1                | HPA opdracht met 1
        BPA   INSTR1           | verhoog het adresgedeelte van de
        HPA   INSTR2           | BPA opdracht met 1
        OPA * 1                |
        BPA   TELLER           |
        OPB * 1                | teller:=teller +1
```

```
BPB   TELLER
AFB * 100          test
SNB   INSTR1
```

Bij de label INSTR1 staat de opdracht HPA LIJST1.

Nu aangenomen dat het invoerprogramma het symbolische adres LIJST1 vertaald heeft in het adres 554 en voorts dat de opdracht HPA intern overeenkomt met het getal 20, dan ziet de instruktie er intern uit zoals figuur A toont.

+	0	0	0	0	0	2	0	0	5	5	4

fig. A

+	0	0	0	0	0	0	0	0	0	0	1

fig. B

+	0	0	0	0	0	2	0	0	5	5	5

fig. C

De drie instrukties

```
HPA   INSTR1
OPA * 1
BPA   INSTR1
```

gaan nu achtereenvolgens het volgende doen.

De eerste van de drie haalt de inhoud van adres INSTR1 naar accu A.
Accu A heeft nu dezelfde inhoud als te zien is in fig.A.
De volgende instruktie telt bij accu A het getal 1 op. Het getal 1 ziet er intern uit zoals fig. B laat zien.
Na de optelling is de inhoud van accu A met 1 verhoogd.Zie fig. C.
De laatste opdracht brengt accu A over naar het adres INSTR1, waardoor op die plaats nu niet meer de opdracht HPA LIJST1 staat, maar :
HPA LIJST1 + 1, want het adres dat overeenkomt met LIJST1 is met 1 verhoogd.

Elke keer dat de lus weer wordt doorlopen, wordt op deze wijze het adres van de HPA instruktie verhoogd.

Uiteraard gebeurt hetzelfde met de BPA instruktie.
Als hetzelfde programma, nadat het al een keer doorlopen is, weer opnieuw wordt gestart, dan geeft dat moeilijkheden. Bij de labels INSTR1 en INSTR2 staat dan immers geen HPA LIJST1 en BPA LIJST2 meer, maar resp. HPA LIJST1 + 100 en BPA LIJST2 + 100.

Hier krijgen we te maken met het belang van initialiseren.

Bij de bovenstaande methode van adressenrekening is het daarom noodzakelijk een iets andere wijze van programmeren toe te passen, waardoor het programma elke keer dat het opnieuw wordt gestart, ook elke keer dezelfde instrukties zal uitvoeren.

De wijze waarop dit moet gebeuren wordt getoond in onderstaand programma.

```
        |  :
        |
        |  HPA *
        |  BPA   TELLER
        |  HPA   HAAL
        |  BPA   INSTR1      voorbereiding
        |  HPA   BRENG
        |  BPA   INSTR2
INSTR1  |  0                 verplaats een woord
INSTR2  |  0
        |  HPA   INSTR1
        |  OPA * 1
        |  BPA   INSTR1      verander de adresgedeelten
        |  HPA   INSTR2      van de instrukties
        |  OPA * 1
        |  BPA   INSTR2
        |  HPB   TELLER
        |  OPB * 1           TELLER:=TELLER +1
        |  BPB   TELLER
        |  AFB * 100
        |  SNB   INSTR1
        |  :
        |
TELLER  |  0
HAAL    |  HPA   LIJST1
BRENG   |  BPA   LIJST2
```

In de voorbereiding wordt uit de adressen HAAL en BRENG de instrukties HPA LIJST1 en BPA LIJST2 gehaald en deze worden op de juiste plaats in het geheugen gebracht.
Tijdens het coderen moet op de plaatsen met labels INSTR1 en INSTR2 iets neer worden geschreven. De label moet als het ware ergens aan hangen. Wat er neer geschreven wordt is niet belangrijk als er maar iets staat.
De rest van het programma is identiek met het vorige.
Elke keer dat het programma herstart wordt, zal op de adressen INSTR1 en INSTR2 als beginwaarde HPA LIJST1 en BPA LIJST2 worden neergeschreven.

Ir. P.A. Tas

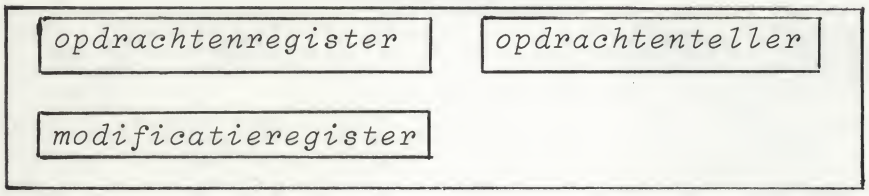
3. MODIFICATIEOPDRACHT

Omdat adresrekening bijzonder vaak voorkomt en de wijze van adresrekening, zoals we in de vorige les hebben behandeld, nogal ingewikkeld is, heeft men een opdracht ontworpen die de naam modificatieopdracht heeft gekregen. *(modificeren betekent veranderen).*
De meeste machines bezitten meerdere modificatieopdrachten. De SERA kent er slechts één.

De modificatieopdracht is een zeer bijzondere opdracht. Hij verandert n.l. de opdracht die op hem volgt.

Om de werking van de modificatieopdracht duidelijk te maken is het noodzakelijk nog eens na te gaan wat er in het besturingsorgaan van de machine gebeurt indien opdracht na opdracht van het programma wordt uitgevoerd.

BESTURINGSORGAAN



Bovenstaand figuur geeft een schets van het besturingsorgaan, met daarin de registers die voor ons hier van belang zijn. Een er van is een nieuw register dat duidelijk te maken heeft met modificatie.

Bij het verwerken van instrukties in het besturingsorgaan worden achtereenvolgens de instrukties vanuit het geheugen gebracht naar het uitvoeringsregister. De opdrachtenteller geeft daarbij aan om welke instruktie het gaat.
Het uitvoeringsregister interpreteert de opdracht en draagt zorg voor de uitvoering ervan, waarna de opdrachtenteller met 1 verhoogd wordt, behalve in het geval van een sprong-instruktie. In het geheugen staat de gehaalde opdracht overigens nog onveranderd.

De spronginstruktie is eigenlijk een instruktie die de opdrachtenteller *forceert naar een andere inhoud.*
Het adresgedeelte van de sprongopdracht wordt de nieuwe inhoud van de opdrachtenteller.
Wat gebeurt er nu als een modificatieopdracht in het besturingsorgaan terecht komt?
Het opdrachtenregister herkent nu aan het operatiegedeelte van de instruktie dat het een modificatieopdracht is.

De uitvoering van deze opdracht komt nu in principe op het

volgende neer :

De inhoud van het woord aangegeven door het adres van de modificatieopdracht, wordt naar het modificatieregister gebracht. Daarna wordt zoals bij de meeste opdrachten de opdrachtenteller met 1 verhoogd en de volgende opdracht opgehaald.
Voordat deze opdracht geïnterpreteerd en uitgevoerd zal worden, wordt de inhoud van het modificatieregister opgeteld bij het opdrachtenregister. Pas dan wordt de inhoud van het opdrachtenregister geïnterpreteerd en de opdracht uitgevoerd.

MOD n Modificeer de volgende opdracht.
De opdracht volgend op de modificatieopdracht wordt in het besturingsorgaan veranderd door bij de inhoud van het opdrachtenregister de inhoud van n op te tellen.

Voorbeeld :

In een programma komen de volgende twee opvolgende instructies voor :

```
    :
50  MOD   100
51  HPA   200
    :
```

De opdrachtenteller staat op 50.
De interne codering van de MOD opdracht is 49 en van de HPA opdracht 20. De inhoud van adres 100 is 36.
Nu wordt de opdracht op adres 50 naar het besturingsorgaan gehaald. De drie registers voor en na de uitvoering van de modificatieopdracht zien er nu uit zoals achtereenvolgens in figuur A en B aangegeven.

uitvoeringsregister

0	0	0	0	0	0	4	9	0	1	0	0

modificatieregister

0	0	0	0	0	0	0	0	0	0	0	0

opdrachtenteller

0	0	5	0

figuur A.

uitvoeringsregister

0	0	0	0	0	0	4	9	0	1	0	0

modificatieregister

0	0	0	0	0	0	0	0	0	0	3	6

opdrachtenteller

0	0	5	1

figuur B.

Dan wordt de volgende opdracht naar het besturingsorgaan gehaald.
Zie figuur C.

Voordat deze opdracht wordt geïnterpreteerd en uitgevoerd, vindt modificatie plaats (figuur D.) en daarna wordt de opdracht in het uitvoeringsregister uitgevoerd.

uitvoeringsregister

0	0	0	0	0	0	2	0	0	2	0	0

modificatieregister

0	0	0	0	0	0	0	0	0	0	3	6

opdrachtenteller

0	0	5	1

figuur C.

Ir. P.A. Tas

uitvoeringsregister

0	0	0	0	0	0	2	0	0	2	3	6

modificatieregister

0	0	0	0	0	0	0	0	0	0	0	0

opdrachtenteller

0	0	5	2

figuur D.

Voorbeelden :

```
MOD   1000
HPA     40
 ¦
```

De inhoud van 1000 = 60.
HPA 40 wordt in het besturingsorgaan veranderd in HPA 40 + (1000) = HPA 40 + 60 = HPA 100.
(1000) betekent : de inhoud van 1000.
De twee opdrachten tezamen zullen hetzelfde doen als de opdracht HPA 100.

```
 ¦
MOD   500 }        (500) = 10
BPB    50 } BPB 50 + (500) = BPB 50 + 10 = BPB 60
```

De inhoud van 500 wordt opgeteld bij de volgende opdracht.
Aangezien (500) = 10 wordt de volgende opdracht gelijk aan BPB 60.

OPMERKING 1.

Hoewel het reeds is opgemerkt, moet er met nadruk op gewezen worden dat de modificatie-opdracht slechts werkt op de volgende opdracht.
Verdere opdrachten worden niet beïnvloed.

OPMERKING 2.

Modificatie wordt in het algemeen slechts toegepast op adres-

sen, hoewel het mogelijk is om het operatiegedeelte van een opdracht te veranderen.

Natuurlijk moeten modificatie-opdrachten ook gebruikt kunnen worden met symbolische adressen.

```
        ¦
        ¦
MOD   TELLING
HPA   LIJST
        ¦
        ¦
```

HPA LIJST zal in het besturingsorgaan geïnterpreteerd worden als :
HPA LIJST + (TELLING)

Het invoerprogramma verwerkt de symbolische namen tot geheugenadressen.

Het voorbeeld van de verplaatsing van een groep woorden in het geugen.

```
        HPA *
        BPA   TELLER      teller:=0
LUS     MOD   TELLER   }
        HPA   LIJST1   }
        MOD   TELLER   }  verplaats een woord
        BPA   LIJST2   }
        HPB   TELLER
        OPB * 1           teller:=teller+1
        BPB   TELLER
        AFB * 100         teller=100?
        SNB   LUS
```

De eerste twee opdrachten maken de teller schoon; teller:=0.
De derde modificeert de vierde opdracht.
De inhoud van TELLER wordt opgeteld bij de volgende instruktie. Aangezien we praktisch altijd de adressen zullen modificeren zal in het vervolg gezegd worden : de inhoud van TELLER wordt opgeteld bij het adres van de volgende opdracht.

4. INDEXGEHEUGEN

Voor het rekenen met adressen is in vele machines een speciaal geheugen aanwezig: indexgeheugen. Dit is een geheugen met een vrij beperkt aantal woorden. De lengte van zo'n woord wordt bepaald door het grootste adres van het geheugen en zou dus, indien aanwezig bij de SERA, bestaan uit 4 tetraden of 16 bits.

Ir. P.A. Tas

In deze indexgeheugenplaatsen kunnen ook tellingen plaats vinden. De programmering voor de machine wordt moeilijker omdat speciale opdrachten vereist zijn; wel wordt de programmering meer flexibel. Vele moderne machines hebben een meeradrescode waarbij een van de adressen betrekking heeft op een indexregister. In dat geval wordt elke opdracht gemodificeerd.
Noodzakelijk is een indexgeheugen niet omdat alle bewerkingen ook in het gewone geheugen kunnen plaatsvinden. De SERA-machine bezit geen indexgeheugen.

5. VOORBEELD

De provisie van een verkoper kan op verschillende manieren berekend worden, afhankelijk van het produkt dat verkocht wordt. De formules voor de berekening zijn zo uiteenlopend, dat zij niet in één formule met verschillende constanten gecomprimeerd kunnen worden.
Er wordt aangenomen dat er 5 formules zijn waaruit gekozen moet worden met behulp van een cijfer 1, 2, 3, 4 of 5 behorende bij één van de vijf formules.
Elk produkt dat verkocht wordt heeft dus een bijbehorende produktcodenummer.

De formules behorende bij de produktcode zien er als volgt uit :

1 0,15 x (verkoopprijs)
2 0,40 x (verkoopprijs - inkoopprijs)
3 0,10 x (inkoopprijs) + 0,50 x (verkoopprijs-inkoopprijs)
4 *f*. 10,-- + 0,05 x (inkoopprijs)
5 *f*. 15,--

De invoer op kaarten moet als volgt geponst worden :

in kolom

1 - 4	nummer van de verkoper
10 - 13	nummer van het produkt
20 - 25	inkoopprijs in centen
30 - 35	verkoopprijs in centen
40	produktcode
50 - 53	aantal verkochte eenheden

Gevraagd wordt een programma te schrijven die per kaart de provisie berekent en deze, met bovenstaande gegevens, afdrukt op de sneldrukker.
Het einde van de verwerking is bereikt als een kaart gelezen wordt met in kolom 1 t/m 4 de letters z.

Ir. P.A. Tas

```
WEER    BGN
        LSK
        BUS   80:120
        HAB   1:4
        HPA   Z                    test op einde
        SAB   EIND
        HAB   20:25
        KAG   6                    inkoopprijs ---> INKOOP
        BPB   INKOOP
        HAB   30:35
        KAG   6                    verkoopprijs ---> VERKOOP
        BPB   VERKOOP
        HAB   50:53
        KAG   4                    aantal eenheden ---> AANTAL
        BPB   AANTAL
        HAB   40:40
        KAG   1                    produktcode ---> ACCU-B
        HPA *
        MOD   ACCU-B
LADDER  SAL   LADDER
        SAL   FORM1
        SAL   FORM2
        SAL   FORM3
        SAL   FORM4
        HPB * 1500                 formule 5
        SAL   PROV
FORM1   HPB   VERKOOP  }
        VMG * 15       }           formule 1
        DLN * 100      }
        SAL   PROV
FORM2   HPB   VERKOOP  }
        AFB   INKOOP   }           formule 2
        VMG * 4        }
        DLN * 10       }
        SAL   PROV
FORM3   HPB   VERKOOP  }
        AFD   INKOOP   }
        DLN * 2        }
        BPB   VERKOOP  }           even wegbergen.De inhoud van
        HPA *          }           VERKOOP wordt niet meer gebruikt
        HPB   INKOOP   }
        DLN * 10       }
        OPB   VERKOOP  }           formule 3
        SAL   PROV
FORM4   HPB   INKOOP   }
        DLN * 20       }           formule 4
        OPB * 1000     }
PROV    VMG   AANTAL               provisie
        KGA   s10                  konverteer in hexaden
        BAB   60:69
```

```
        |  DRU
        |  SAL    WEER
EIND    |  STP
Z       |  "OOOOZZZZ"
INKOOP  |  0
VERKOOP |  0
AANTAL  |  0
        |  SRT
```

Met behulp van de juiste produktcode en de modificatie-opdracht wordt de juiste sprongopdracht gekozen die op zijn beurt naar het stukje programma springt behorende bij die produktcode. De achter elkaar geprogrammeerde onvoorwaardelijke sprongen vormen samen een *"LADDER"*.

6. MEERVOUDIGE ADRESMODIFICATIES

Bij meer ingewikkelde problemen zal het meervoudig modificeren van een opdracht vaak voorkomen.
De vorm die dan gekozen wordt is echter met onze modifikatie-opdracht niet zo eenvoudig te verwezenlijken.
Meestal zal het zo zijn dat het adresgedeelte van een opdracht veranderd zal moeten worden, door er de inhoud van 2 of meerdere geheugenplaatsen bij op te tellen.

Bijvoorbeeld : tel de inhoud van 100 en de inhoud van 101 op bij het adres van de opdracht HPA 200.

(100) = 50, (101) = 60.
Dus HPA 200+(100)+(101)=
=HPA 200+50+60=
=HPA 310

Dit is niet op te lossen door twee modificatie-opdrachten achter elkaar te plaatsen, gevolgd door de haalopdracht, want :

```
MOD   100 }          { MOD   101+(100) }
MOD   101 }    =     { HPA   200       }  =
HPA   200 }

{ MOD   151 }
{ HPA   200 }    =    HPA  200+(151)
```

Een andere methode moet dan gevolgd worden :

```
HPA   100     }  (100)+(101)=50+60=110      ACCU-A
OPA   101     }
MOD   ACCU-A  }= HPA 200+(A)=HPA   200+110=HPA  310
HPA   200     }
```

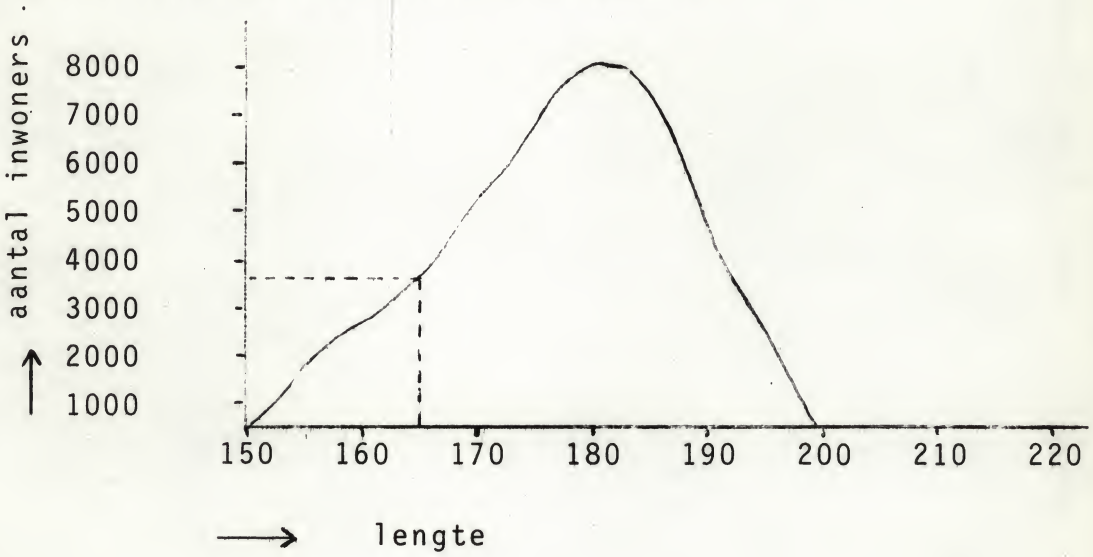
Overzicht van het KLM computercentrum te Amstelveen

Er zijn wel problemen waar het gebruik van meerdere modifikatie-opdrachten achter elkaar nuttig zijn.

7. NU VOLGEN EEN AANTAL VOORBEELDEN.

Voorbeeld 1

In een dorp wordt van alle inwoners boven de 20 jaar de lengte gemeten. Men is geïnteresseerd in een verdeling die aangeeft hoeveel mensen een bepaalde lengte bezitten.
Er wordt in centimeters gemeten. De min. en max. lengte zijn resp. 150 en 220 cm.
Het gaat dus om de getallen die aangeven hoeveel mensen 150, 151, 152, 153 ..... 220 cm. lang zijn.
Zijn deze getallen bekend, dan zou men die in een grafiek kunnen uitbeelden zoals bijv. in figuur 1.



figuur 1.

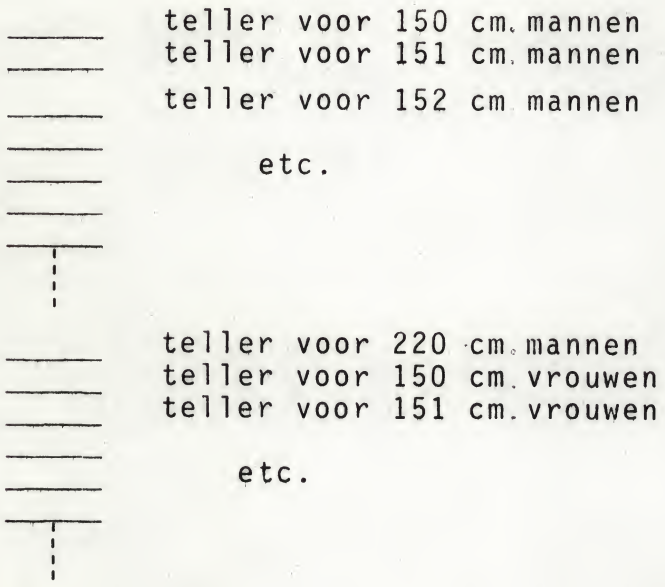
De lengtes zijn horizontaal afgezet.
De aantallen staan op de vertikale as.
Het aantal mensen met een lengte van bijv. 165 cm. bedraagt in deze grafiek ongeveer 3650.

De gegevens zijn verzameld op kaarten (aantal onbekend).
De laatste kaart is een sluitkaart waarop de letter z in kolom 1.
Elke kaart bevat :

in kolom 2-4 een getal dat de lengte aangeeft
in kolom 1 de letter m, v of z.

m en v geven resp. aan dat een man of vrouw is gemeten.

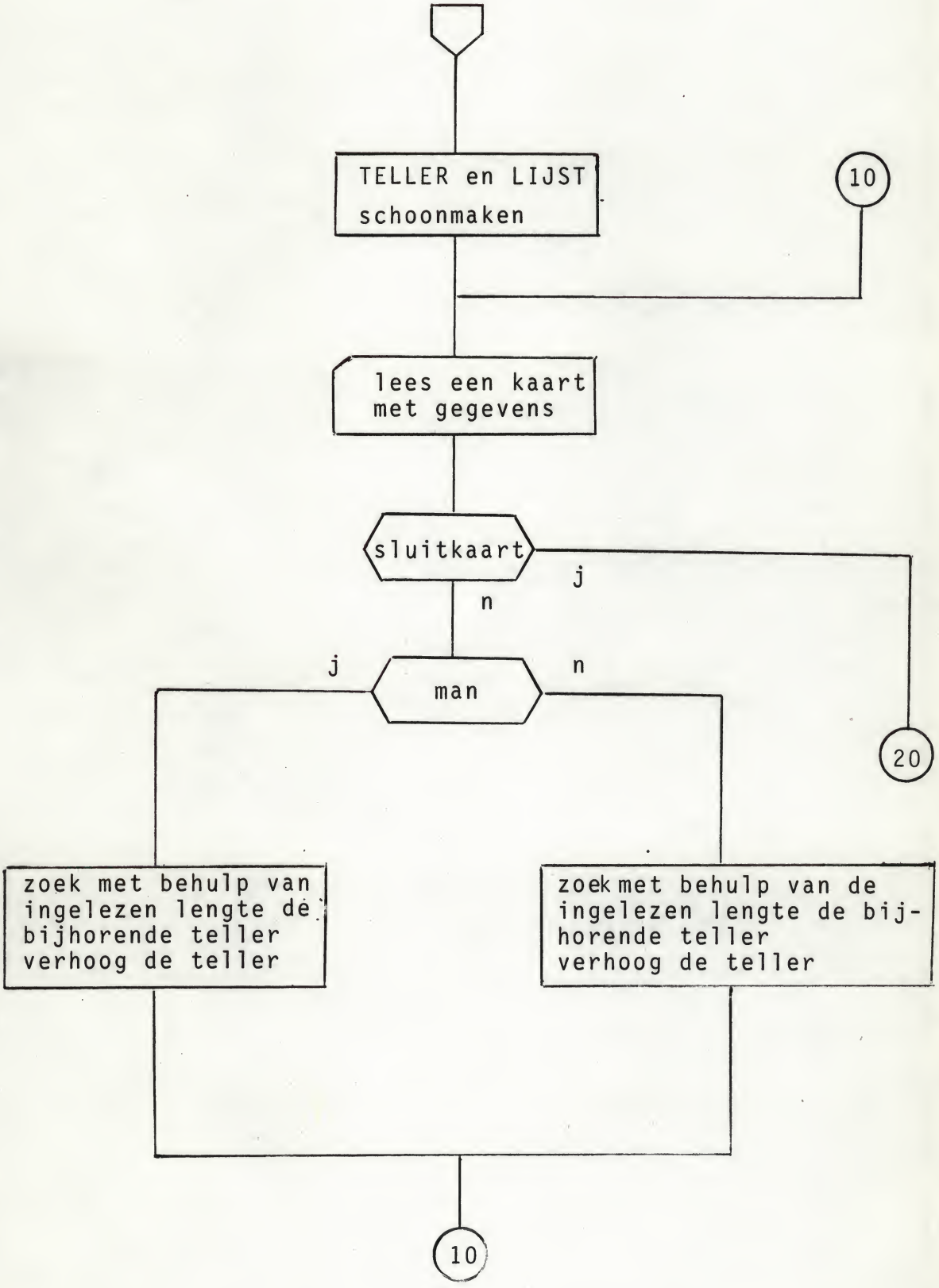
Er worden 3 verdelingen gevraagd : één voor mannen, één voor vrouwen en één voor beiden samen. Bovendien is het aantal gemeten inwoners van belang.
Het aantal inwoners van het dorp bedraagt ongeveer 20.000, zodat het aantal mensen van een bepaalde lengte nooit groter dan een getal van 5 cijfers kan zijn.
Er zijn 220 - 150 +1 = 71 lengten die in aanmerking komen.
Dit betekent dat er 2x 71 = 142 tellers nodig zijn waarin moet worden "geturfd".
De tellers worden verzameld in een lijst, die de label LYST krijgt. De tellers voor de mannen hebben de adressen LYST t/m LYST+70.
De vrouwen worden geteld in LYST+71 t/m 141.
Mannen die 220 cm lang zijn worden bijv. geteld in LYST+70.
LYST+71 is de teller voor de vrouwen die 150 cm lang zijn etc.
Zie figuur 2.

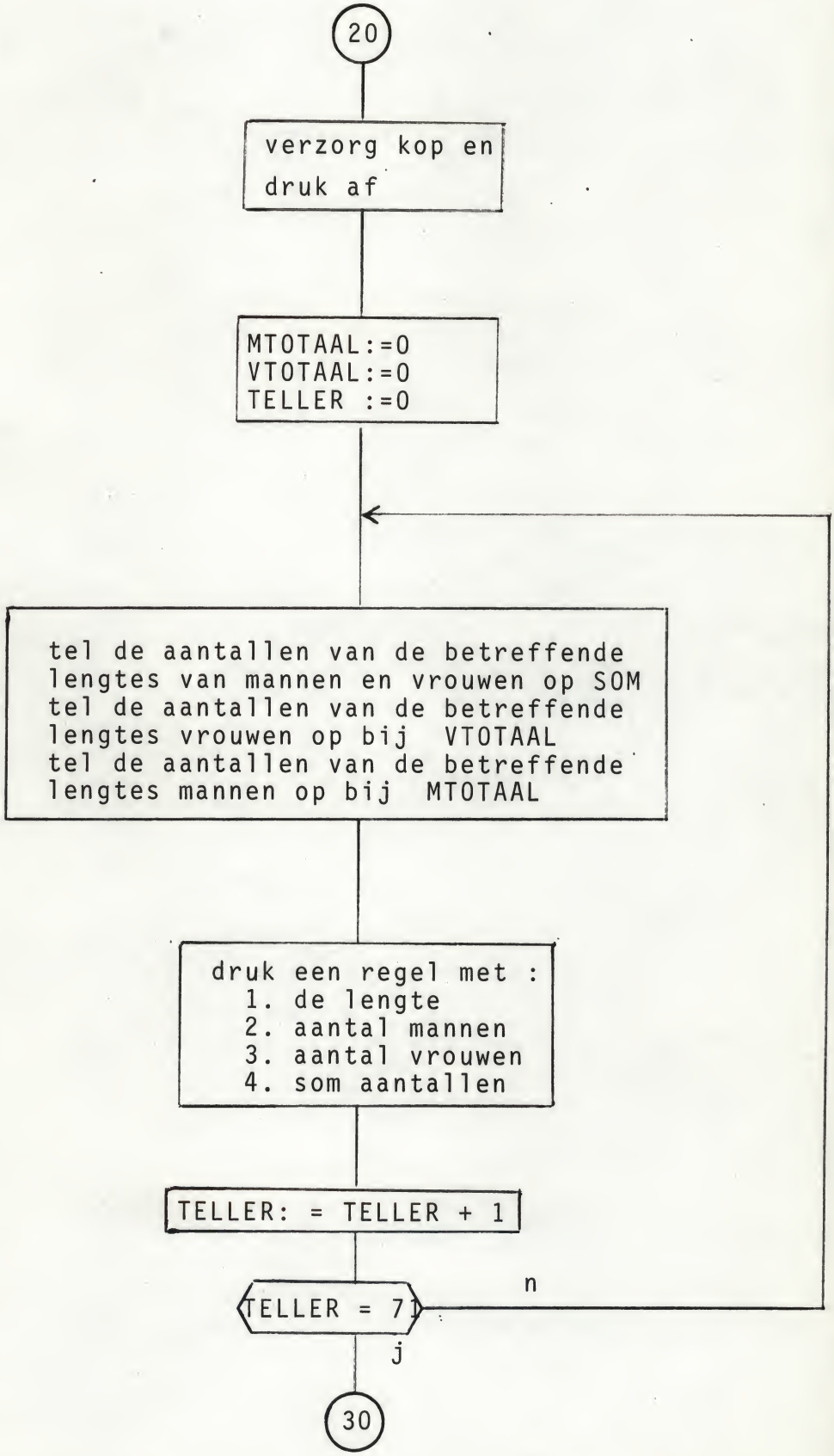


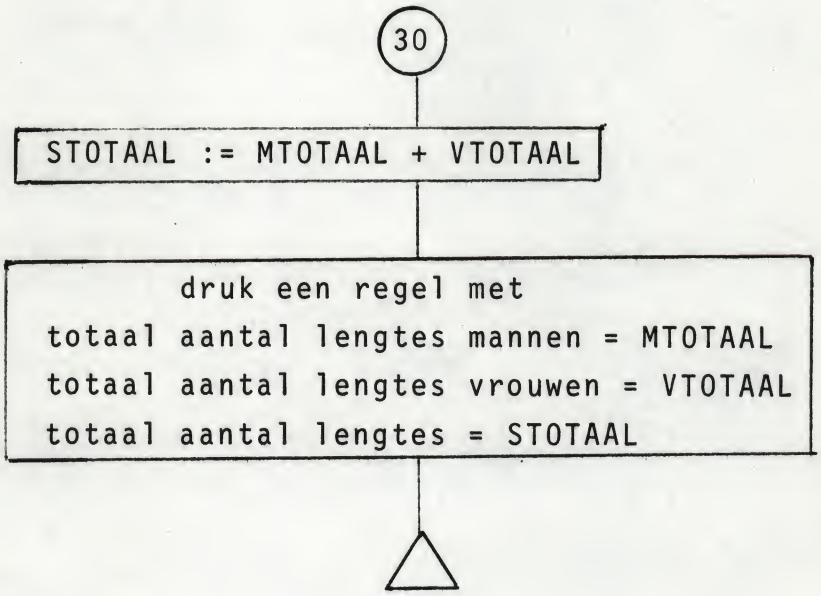
figuur 2.

Met behulp van de lengte die wordt ingelezen, moet de juiste letter worden gekozen, waarvan de inhoud met 1 wordt verhoogd.
Is de ingelezen lengte bijv. 152 cm en werd er een man gemeten dan wordt de inhoud van LYST + 2 verhoogd.
Is de gelezen lengte 153 cm, dan wordt LYST + 3 verhoogd.
Op de één of andere manier moet er een overeenkomst zijn tussen de lengte en de bijbehorende teller. Dit zal straks bij het programma nader worden bekenen.
Als er nog geen kaart is ingelezen, kan er ook nog niet zijn geteld. Dit betekent dat bij de initialisering de inhoud van alle tellers gelijk aan 0 moet zijn.

Ir. P.A. Tas


TELLER en LIJST schoonmaken
10
lees een kaart met gegevens
sluitkaart
j
n
j
man
n
20
zoek met behulp van ingelezen lengte de bijhorende teller verhoog de teller
zoek met behulp van de ingelezen lengte de bijhorende teller verhoog de teller
10


20
verzorg kop en
druk af
MTOTAAL:=0
VTOTAAL:=0
TELLER :=0
tel de aantallen van de betreffende
lengtes van mannen en vrouwen op SOM
tel de aantallen van de betreffende
lengtes vrouwen op bij VTOTAAL
tel de aantallen van de betreffende
lengtes mannen op bij MTOTAAL
druk een regel met :
1. de lengte
2. aantal mannen
3. aantal vrouwen
4. som aantallen
TELLER: = TELLER + 1
TELLER = 71
n
j
30



Een verklaring van de gebruikte namen :

LYST	**reeds besproken.**
TELLER	**deze naam spreekt voor zichzelf.**
MTOTAAL	hierin wordt het totaal aantal gemeten mannen geteld.
VTOTAAL	hetzelfde als boven, maar nu voor vrouwen.
STOTAAL	totaal aantal gemeten personen.
SOM	een hulpwoord.

HET PROGRAMMA

```
        BGN
        BSA   TELLER
        BPA   TELLER        0 ---> TELLER
WEER    MOD   TELLER  )
        BPA   LYST     )
        HPB   TELLER   )
        OPB * 1        )    schoonmaken van de lijst.
        BPB   TELLER   )
        AFB * 142      )
        SNB   WEER     )
LEES    LSK
        HAB   1:1           1e kolom ---> B
        HPA   TEKST+5       hexade z ---> A
        SAB   VOORBER       spring (A)=(B)
```

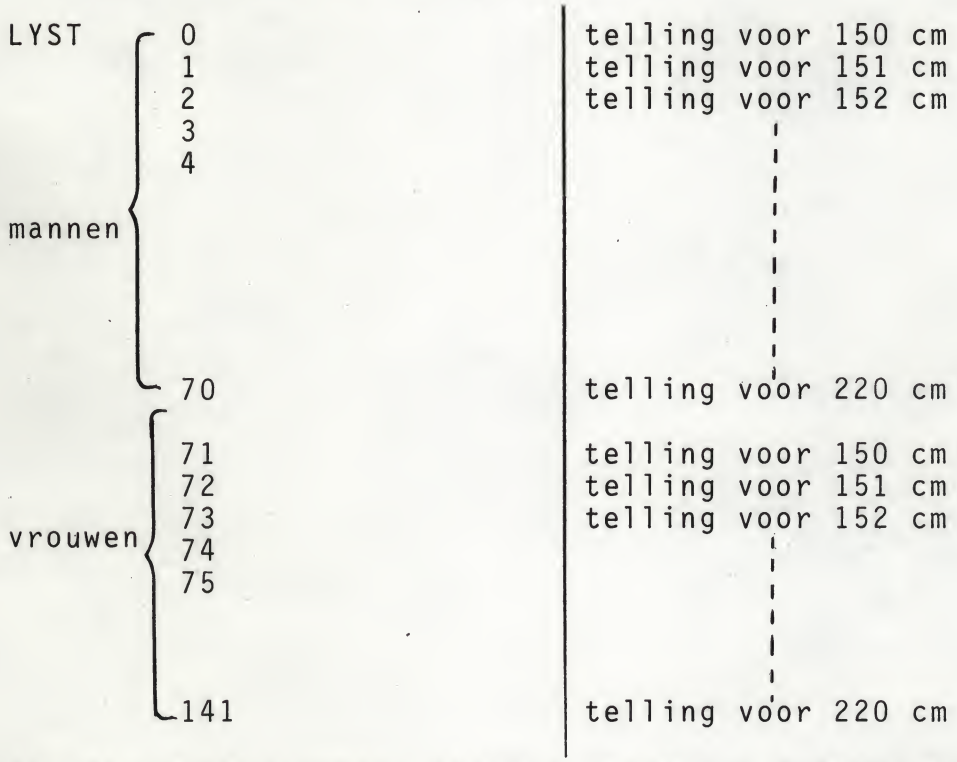
Ir. P.A. Tas

```
        HAB    2:4          lengte ⟶ B
        KAG    3            konverteer
        BPB    LENGTE
        HAB    1:1
        HPA    TEKST+6      hexade m ⟶ B
        SAB    MAN
        HPA    LENGTE
        OPA  * 71
        BPA    LENGTE
MAN     MOD    LENGTE
        HPA    LYST-150
        OPA  * 1            tel 1 op bij de teller behoren-
        MOD    LENGTE       de bij de betreffende lengte.
        BPA    LYST-150
        SAL    LEES
VOORBER BUS    1:120        buffer schoon
        HPB    TEKST
        BAB    6:11
        HPB    TEKST+1      verzorg opmaak en druk af
        BAB    21:26
        HPB    TEKST+2
        BAB    36:42
        HPB    TEKST+3
        BAB    52:56
        DRU
        DRN    4
        BSA    TELLER
        BPA    TELLER       +0 ⟶ TELLER
        BPA    MTOTAAL      +0 ⟶ MTOTAAL
        BPA    VTOTAAL      +0 ⟶ VTOTAAL
TELLING MOD    TELLER       De aantallen van overeenkomstige
        HPA    LYST         lengten worden opgeteld.
        MOD    TELLER
        OPA    LYST+71
        BPA    SOM
        HPA    MTOTAAL      het maken van een totaal voor
        MOD    TELLER       mannen
        OPA    LYST
        BPA    MTOTAAL
        HPA    VTOTAAL
        MOD    TELLER       het maken van een totaal voor
        OPA    LYST+71      vrouwen
        BPA    VTOTAAL
        MOD    TELLER
        HPB  * 150
        KGA    s3           verzorg de lengte
        BAB    9:11
        MOD    TELLER
        HPB    LYST         aantal mannen ⟶ buffer
        KGA    s5
        BAB    22:26
```

```
          MOD   TELLER
          HPB   LYST+71
          KGA   s5          aantal vrouwen ──→ buffer
          BAB   38:42
          HBP   SOM
          KGA   s5          gezamelijk aantal ──→ buffer
          BAB   52:56
          DRU               druk af
          DRN   1
          HPA   TELLER
          OPA * 1           TELLER:=TELLER+1
          BPA   TELLER
          AFA * 71
          SNA   TELLING
          HPA   MTOTAAL
          OPA   VTOTAAL     totaal samen maken
          BPA   STOTAAL
          BUS   1:120
          DRN   3
          HPB   TEKST+4
          BAB   15:20
          BAB   31:36
          BAB   45:50
          HPB   MTOTAAL
          KGA   s5          totalen afdrukken
          BAB   22:26
          HPB   VTOTAAL
          KGA   s5
          BAB   38:42
          HPB   STOTAAL
          KGA   s5
          BAB   52:56
          DRU
          DRM   10          10 nieuwe regels
          STP
HULP      ----  0
LENGTE          0
TELLER          0
MTOTAAL         0
VTOTAAL         0
STOTAAL         0
SOM             0
TEKST           "lengte"
                "mannen"
                "vrouwen"
                "samen"
                "totaal"
                "0000000Z"
                "0000000M"
LYST      BGN   LYST+141
          SRT
```

De modificatie-opdracht wordt in dit voorbeeld op twee verschillende manieren toegepast.

1. Door een ingelezen grootheid wordt bestemd in welke geheugenplaats er geteld moet worden. Het is bekend dat de lengten die voorkomen tussen 150 cm en 220 cm liggen. Er zijn 2 lijsten, één voor mannen en één voor vrouwen die achter elkaar in het geheugen staan.



In elk woord van deze lijsten wordt een telling bijgehouden, die aangeeft het aantal mannen of vrouwen van een bepaalde lengte.
Er wordt bij voorbeeld de lengte 152 ingelezen en in de geheugenplaats LENGTE gebracht.
Stel verder dat in de eerste kolom van de kaart een m staat.
Met behulp van MOD LENGTE wordt de teller voor 152 cm gevonden.

```
MOD     LENGTE
HPA     LYST-150   = HPA   LYST - 150+(LENGTE)
                   = HPA   LYST - 150+152
                   = HPA   LYST + 2
```

Zou de lengte behoren bij een vrouw, dan wordt eerst bij 152 het getal 71 opgeteld.
Door de zelfde 2 opdrachten komt er nu :

```
HPA  LYST-150+(LENGTE)=HPA  LYST-150+152+71=HPA  LYST+73
```

LYST+73 is de teller voor vrouwen die 152 cm lang zijn.

2. Met behulp van de TELLER worden grootheden die achter elkaar in het geheugen staan, gehaald en verwerkt.
Verder wordt in het begin van het programma met behulp van TELLER de tellijst schoon gemaakt.
Het gebruik van de modificatie-opdracht in combinatie met een teller, komt bijzonder veel voor.

Het stuk programma dat ligt tussen de labels LEES en VOORBER doet het rekenwerk. De rest van het programma dient hoofdzakelijk voor de uitvoer.
Na de label VOORBER komt de verzorging van de kop en het schoonmaken van een teller en 2 geheugenwoorden waarin totalen van gemeten mannen en vrouwen worden geteld.
Deze telling geschiedt na de label TELLING.
De tellers van de overeenkomstige lengten van mannen en vrouwen worden gevonden door op de juiste manier het adres LYST met een teller te modificeren. Deze lengten worden opgeteld en naar SOM gebracht, zodat het totaal aantal gemeten personen van een lengte nu bekend is. Daarna wordt er bijgeteld in MTOTAAL en VTOTAAL.
Nu volgt het afdrukken, waarbij de betreffende lengte die afgedrukt moet worden ontstaat, door het getal 150 te modificeren met de TELLER.
Als alle lengten aan de beurt zijn geweest worden de totalen afgedrukt.

Het stroomdiagram dat voor dit probleem is opgesteld, munt niet uit door duidelijkheid. Er worden te weinig symbolische namen in gebruikt en de speciale toepassing van de adresmodificatie valt er niet uit te lezen. Dit laatste bezwaar kan worden opgeheven door een schrijfwijze te hanteren die wordt toegepast in de programmeertaal ALGOL.

Veronderstel dat LYST een label is die toegevoegd is aan een groep woorden in het geheugen.

Men kan de inhoud van deze woorden bereiken, door gebruik te maken van de symbolische schrijfwijze :

LYST, LYST+1, LYST+2, LYST+3 etc.

Wil men bijv. het derde woord naar ACCU-A halen dan kan dat gebeuren door te schrijven : HPA LYST+2
Het tiende woord kan worden gehaald door de instruktie:

HPA LYST+9

Het n^e woord wordt gehaald door de instruktie :

HPA LYST+n-1

Het is ook mogelijk om te refereren naar een plaats in het geheugen die voor de label LYST ligt.
Men kan dus ook schrijven : HPA LYST-5.
In dit geval wordt de inhoud gehaald van het woord dat 5 plaatsen voor de label LYST ligt.

Er wordt nu een schrijfwijze ingevoerd, die de volgende structuur heeft :

LYST [N]

N kan alle positieve en negatieve waarden en 0 aannemen, dus
.....-4, -3, -2, -1, 0, 1, 2, 3, 4, ...

Wordt nu LYST [0] geschreven dan heeft dat betrekking op de geheugen-plaats LYST.
LYST [5] duidt op de geheugenplaats LYST+5 en in het algemeen kan gezegd worden dat LYST [N] betrekking heeft op de geheugenplaats LYST+N.

INDEX.

Datgene wat tussen de vierkante haken staat wordt index genoemd en bepaald welke geheugenplaats ten opzichte van LYST gewenst is.

Voorbeeld.

LYST 1 heeft betrekking op de geheugenplaats LYST+1
LYST 10 " " " " " LYST+10
LYST-3 " " " " " LYST-3

Maar ook is het zo dat LYST [TELLER] betrekking heeft op de geheugenplaats LYST+(TELLER).
Dit laatste kan ook worden opgevat als het symbolische adres LYST gemodificeerd met TELLER.

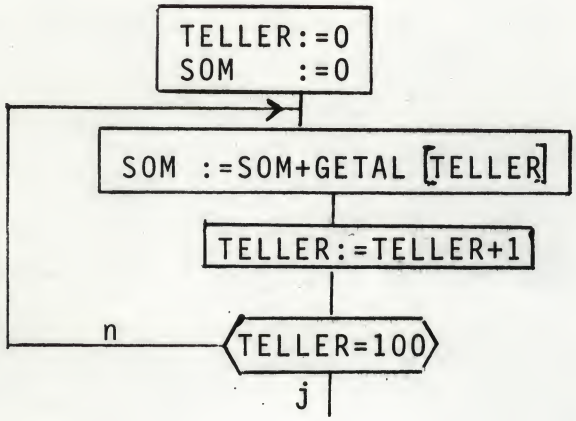
Ir. P.A. Tas

Stel dat de inhoud van TELLER gelijk is aan 30, dan geldt LYST [TELLER] ⟶ LYST [30] en dat wil zeggen : het 31e woord van de lijst.

Men kan de schrijfwijze met de rechte haken gaan toepassen in een stroomdiagram om aan te geven wanneer en hoe gemodificeerd kan worden.

Voorbeeld.

In het geheugen staan, te beginnen bij label GETAL, 100 getallen. Tel deze getallen bij elkaar op.
Het stroomdiagram kan er als volgt uitzien :



Waar het hier natuurlijk om gaat is de uitdrukking :

SOM:=SOM+GETAL [TELLER]

De nieuwe SOM wordt gelijk aan de oude SOM en daarbij opgeteld GETAL [TELLER], d.w.z. er wordt bij opgeteld de inhoud van het symbolische adres GETAL, gemodificeerd met de inhoud van TELLER.
De eerste keer dat de lus wordt doorlopen is TELLER=0.
Bij de som wordt nu opgeteld de inhoud van het geheugenadres GETAL+0 en dat is het eerste getal van de reeks.
Daarna wordt de TELLER verhoogd en wordt de SOM vermeerderd met de inhoud van GETAL+1, het tweede getal etc.

De index mag ook een rekenkundige uitdrukking zijn, dus bijv.
LYST [AxBxC]
LYST [TELLER+50]
LYST [4+TELLER-PIET]
LYST [AANTAL/4]

De uitdrukking tussen de vierkante haken wordt eerst berekend. De uitkomst daarvan bepaalt welke geheugenplaats ten opzichte van LYST wordt bedoeld.

Voorbeeld

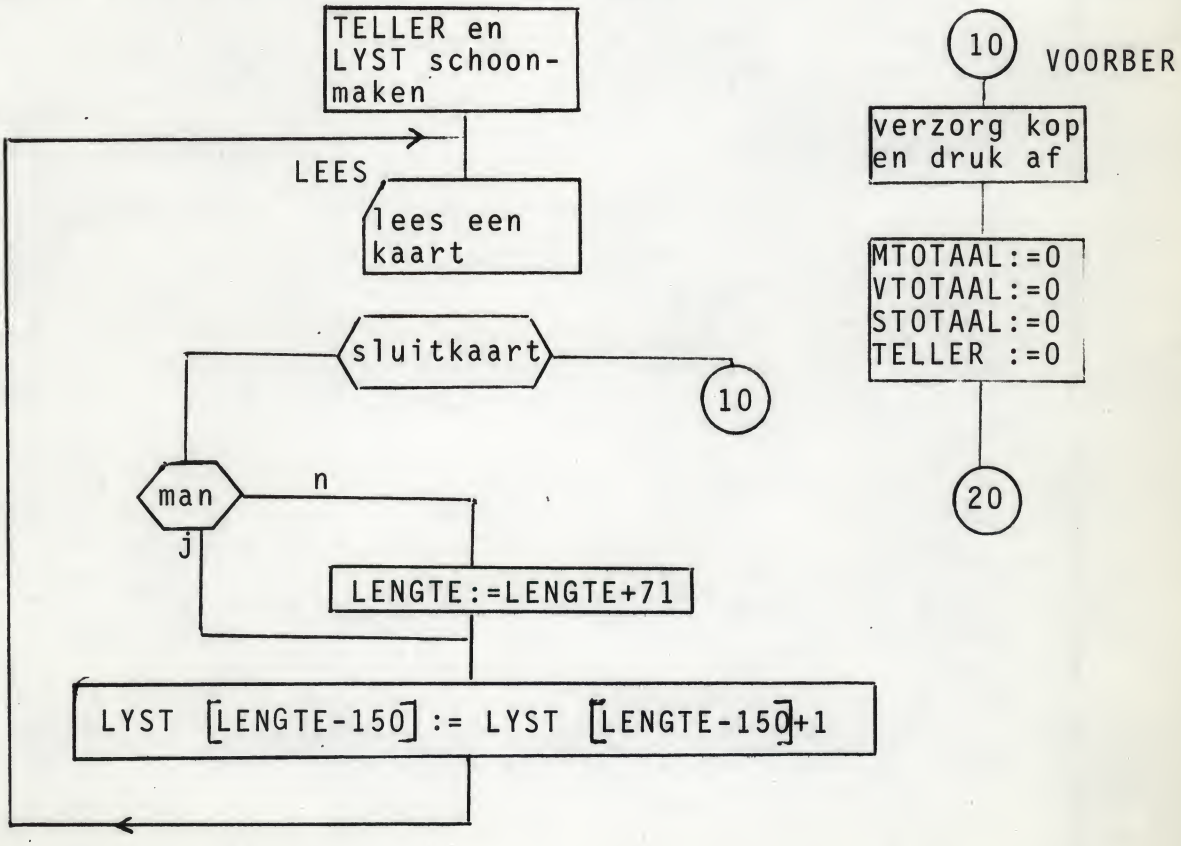
de geheugenplaats met label	heeft de waarde
AANTAL	25
TELLER	2
AF	10

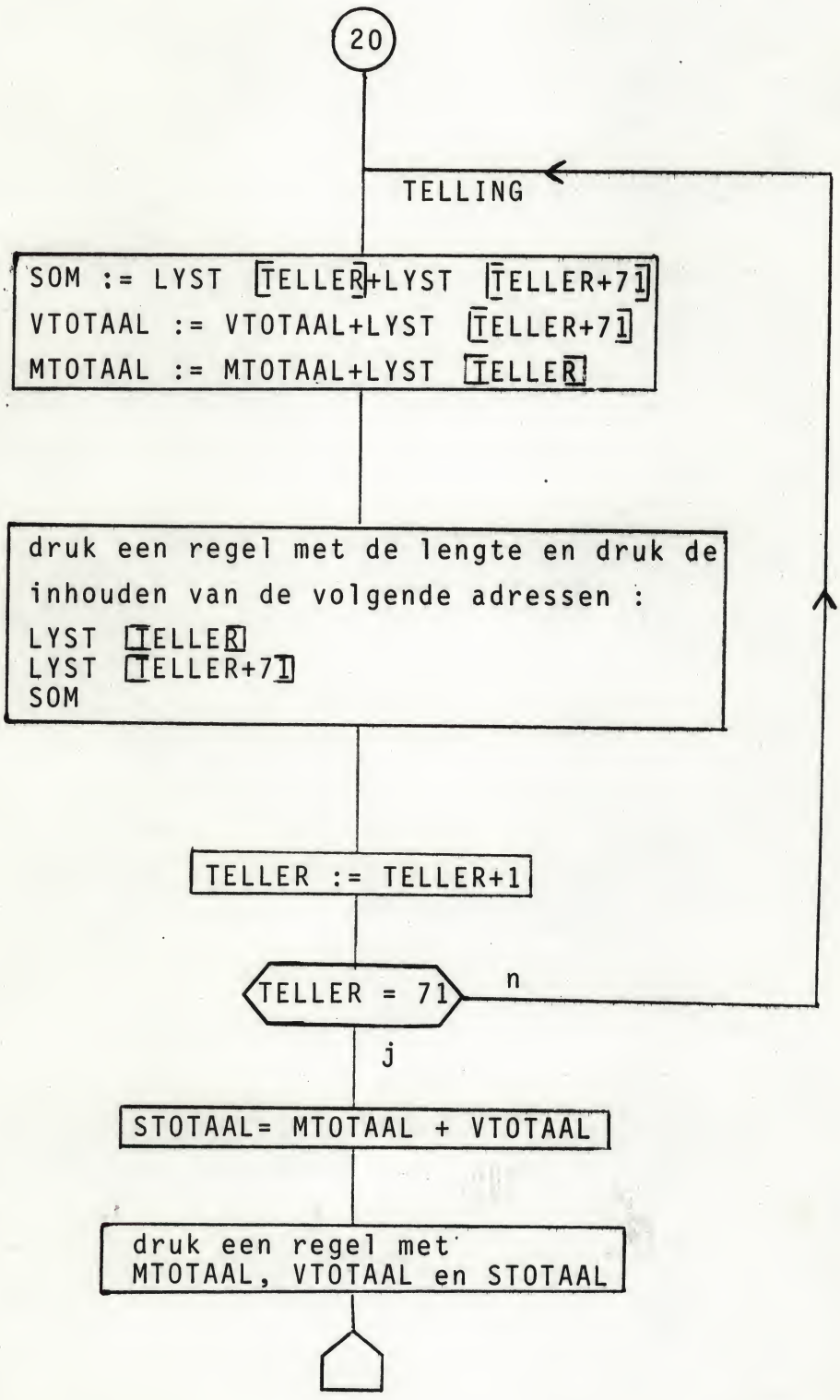
LYST [TELLERxAANTAL-AF] =
LYST [25x2-10] = LYST [40]

Er wordt hier dus verwezen naar LYST+40 en dat is het 41e element van LYST.

In het vervolg zal, waar dit nodig is, gebruik worden gemaakt van bovenstaande schrijfwijze.

Allereerst zal van voorbeeld 1 een geheel ander en meer overzichtelijk stroomdiagram worden opgezet.




20
TELLING
SOM := LYST [TELLER]+LYST [TELLER+71]
VTOTAAL := VTOTAAL+LYST [TELLER+71]
MTOTAAL := MTOTAAL+LYST [TELLER]
druk een regel met de lengte en druk de
inhouden van de volgende adressen :
LYST [TELLER]
LYST [TELLER+71]
SOM
TELLER := TELLER+1
TELLER = 71
n
j
STOTAAL= MTOTAAL + VTOTAAL
druk een regel met
MTOTAAL, VTOTAAL en STOTAAL

Verklaringen bij het stroomdiagram.

Aan de hand van voorbeelden zal de index schrijfwijze van dit stroomdiagram duidelijk worden gemaakt.

a. Er wordt een kaart gelezen met in de eerste kolom een m en lengte 193. LENGTE := 193.
Het is een man waarvan de lengte gemeten is.
Nu wordt het volgende uitgevoerd:
LYST [LENGTE-150] := LYST [LENGTE-150] +1
en dit komt overeen met:
LYST [193-150] := LYST [193-150] +1.
LYST [43] := LYST [43] +1.

Dit laatste betekent: Bij de inhoud van geheugenadres LYST +43 wordt 1 opgeteld en het resultaat wordt weggebracht naar geheugenadres LYST+43.
LYST+43 is de telling voor mannen van de lengte 193 cm.

b. Er wordt een kaart gelezen met in de eerste kolom een v en lengte 168 cm. LENGTE := 168.
Aangezien het een vrouw is, wordt bij de lengte 71 opgeteld.
LENGTE:=LENGTE+71,dus
LENGTE:=168+71 ⟶ LENGTE:=239
Nu volgt weer :
LYST [LENGTE-150] :=LYST [LENGTE-150] +1
LYST [89] :=LYST [89] +1
LYST+89 is de telling voor vrouwen van de lengte 168 cm.

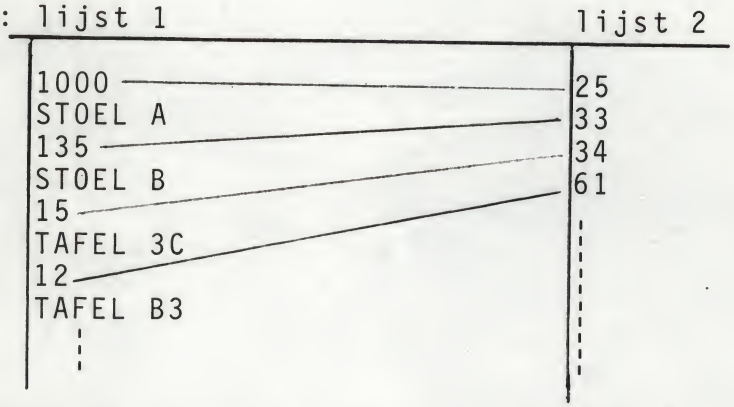
c. In de 1e kolom van de gelezen kaart staat een z.
Een sluitkaart dus. Na het schoonmaken van de TELLER en de totalen, zal de eerste keer het volgende gebeuren.
SOM:=LYST [0] + LYST [0+71]
De inhoud van LYST wordt opgeteld bij de inhoud van LYST +71 en het resultaat wordt naar SOM gebracht.
Dit zijn de tellingen van 150 cm.
VTOTAAL := VTOTAAL + LYST [0+71]
Bij VTOTAAL wordt de telling van het aantal vrouwen van 150 cm. opgeteld.
Iets dergelijks gebeurt voor de mannen.
Na het afdrukken van de regel wordt TELLER met 1 verhoogd.
TELLER:=0+1 ⟶ TELLER:=1
Er wordt getest. TELLER=71 dus er volgt een terugsprong in de lus.
SOM:=LYST [1] + LYST [1+71] ⟶
De inhoud van adres LYST+1 wordt opgeteld bij de inhoud van LYST+72 en het resultaat gaat naar SOM.
Dit zijn de tellingen van 151 cm.

etc.

Voorbeeld 2

Bij de voorraadadministratie is het noodzakelijk om een voorraadlijst bij te houden met daarin aantallen van bepaalde artikelnummers. In dit voorbeeld worden 2 lijsten gebruikt.
Lijst 1, waarin de hoeveelheid en de naam van het art. vermeld staan. Hiervoor zijn 2 woorden per art. nodig.
Lijst 2, die de aanwezige artikelnummers bevat, opvolgend gerangschikt in de geheugenplaatsen en overeenkomend met lijst 1.

Bijvoorbeeld :



De artikelnummers in lijst 2 lopen niet op met 1, omdat een aantal artikelen niet meer wordt gevoerd.
Er worden kaarten met gegevens ingevoerd die de voorraadlijst bijwerken. Het programma signaleert als de voorraad onder 10 stuks daalt. Op de kaart staat aangegeven :

in kolom	
4 - 7	artikelnummer
10 - 17	naam van het artikel
20 - 24	aantal met teken

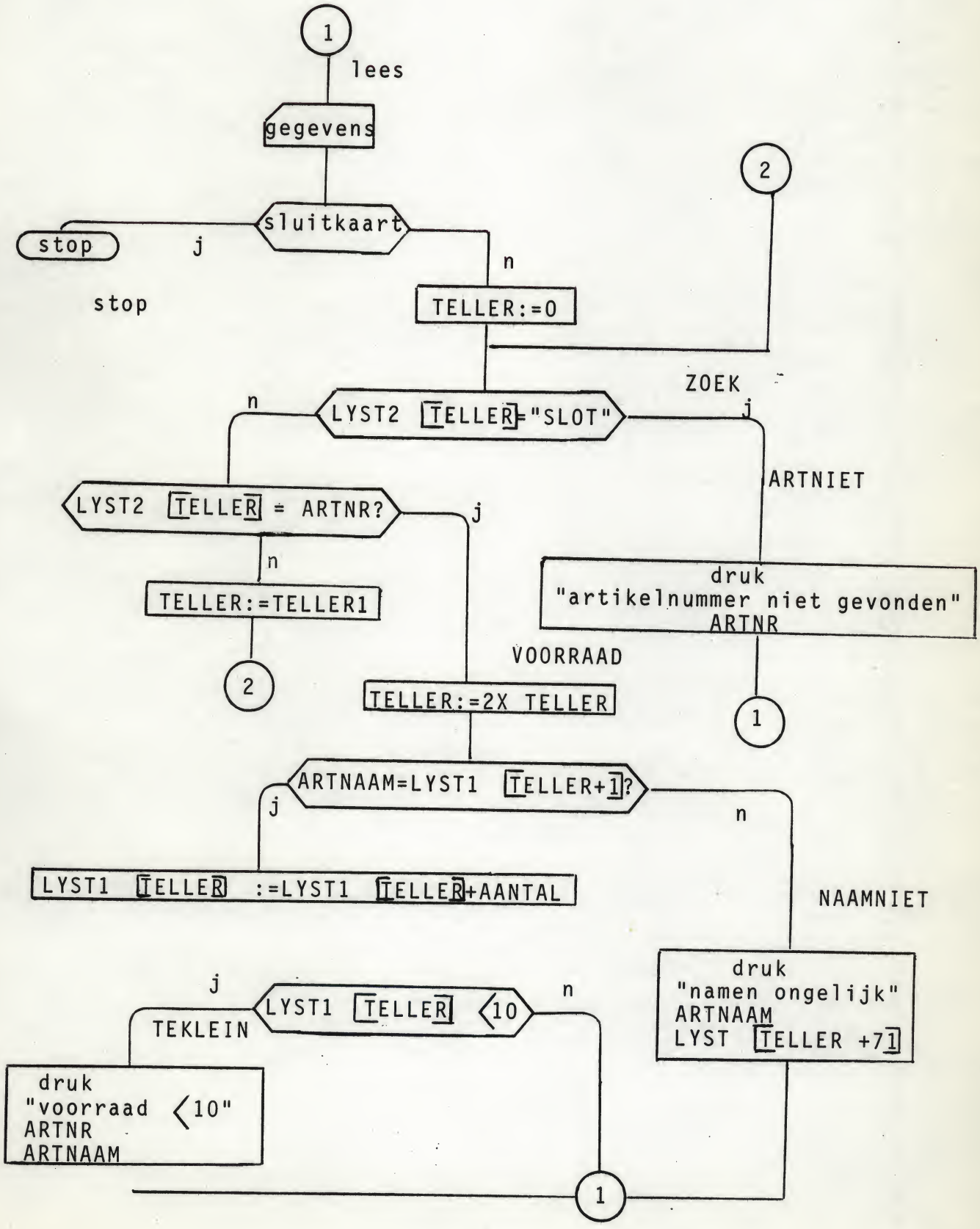
De voorraad wordt aangevuld door het teken van een aantal positief te maken. Opnemen uit de voorraad wordt met een minteken aangegeven.

In het programma worden onder andere de volgende namen gebruikt:

ARTNR	artikelnummer
ARTNAAM	naam van het artikel
AANTA	aantal
LIJST1	
LIJST2	

LYST1 en LYST2 worden met het programma mee in het geheugen gelezen. Het laatste woord van LYST2 bevat de alfanumerieke tekst "SLOT".

De sluitkaart heeft in kolom 4 t/m 7 het woord EIND geponst.



Ir. P.A. Tas

```
        BGN
        BUS    80:120
LEES    LSK
        HAB    4:7
        HPA    TEKST
        SAB    STOP
        BSA    TELLER          +0 ---> A
        BPA    TELLER          +0 ---> TELLER
                               art.nr. ---> ARTNR
        BPA    ARTNR
ZOEK    MOD    TELLER
        HPA    LYST2
        HPB    TEKST+10        test op einde van LYST2
        SAB    ARTNIET
        HPB    ARTNR
        SAB    VOORRAAD        testen op gelijke art.nummers
        HPA    TELLER
        OPA *  1               TELLER:=TELLER+1
        BPA    TELLER
        SAL    ZOEK
VOORRAAD HPB   TELLER
        VMG *  2               TELLER:=2xTELLER
        BPB    TELLER
        HAB    10:17           naam van het art. ---> ACCU B
        MOD    TELLER
        HPA    LYST+1
        SAB    OVER
        SAL    NAAMNIET
OVER    HAB    20:24
        KAG    5               aantal ---> ACCU-B
        MOD    TELLER
        OPB    LYST1           voorraad wordt veranderd
        MOD    TELLER
        BPB    LYST1
        AFB *  10              is de voorraad <10?
        SNB    TEKLEIN
        SAL    LEES
ARTNIET HAB    4:7
        BAB    42:45           breng art. nr. naar positie 42-45
        BUS    1:24            maak de eerste 24 posities schoon
        HPA    TEKST+1
        HPB    TEKST+2
        BAB    10:25           op de sneldrukker de tekst:
        HPA    TEKST+3         ARTIKELNUMMER NIET GEVONDENxxxx
        HPB    TEKST+4
        BAB    26:41
        DRU
        SAL    LEES
```

Ir. P.A. Tas

```
NAAMNIET  HAB    10:17
          BAB    26:33
          BUS    1:10
          HPA    TEKST+5
          HPB    TEKST+6
          BAB    10:25
          MOD    TELLER
          HPB    LYST+1
          BAB    36:43
          DRU
          SAL    LEES
TEKLEIN   HAB    4:7
          BAB    37:40
          BUS    1:10
          HAB    10:17
          BAB    44:51
          HPA    TEKST+7
          HPB    TEKST+8
          BAB    10:25
          HPB    TEKST+9
          BAB    26:33
          DRU
          SAL    LEES
STOP      STP
TEKST     "OOOOEIND"
          "ARTIKELN"
          "UMMER NI"
          "ET GEVON"
          "DEN     "
          "NAMEN ON"
          "GELIJK  "
          "VOORRAAD"
          "KLEINER "
          "DAN 10  "
          "    SLOT"
ARTNR     0
TELLER    0
LYST1     ----------
          ----------
          ----------
               ¦
LYST2     ----------
          ----------
          ----------
               ¦
               ¦
          "    SLOT"
          SRT
```

op de sneldrukker de tekst:

NAMEN ONGELIJK xxxxxxxx xxxxxxxx

op de sneldrukker de tekst:
VOORRAAD KLEINER DAN 10 xxxx xxxxxx

Ir. P.A. Tas

Er zijn vele methoden om voorraadadministratie te voeren.
Dit voorbeeld pretendeert niet de juiste gevolgd te hebben.
Het wil slechts tonen hoe met behulp van de modificatieopdracht in een lijst gezocht kan worden (zie vanaf label ZOEK) en hoe, indien het artikelnummer gevonden is, direkt de betreffende voorraad in een andere lijst gevonden wordt.
Verder zijn enige geprogrammeerde controles ingebouwd.
De methode kan aanzienlijk versneld worden door de in te voeren kaarten van te voren op opklimmend art. nr. te sorteren.

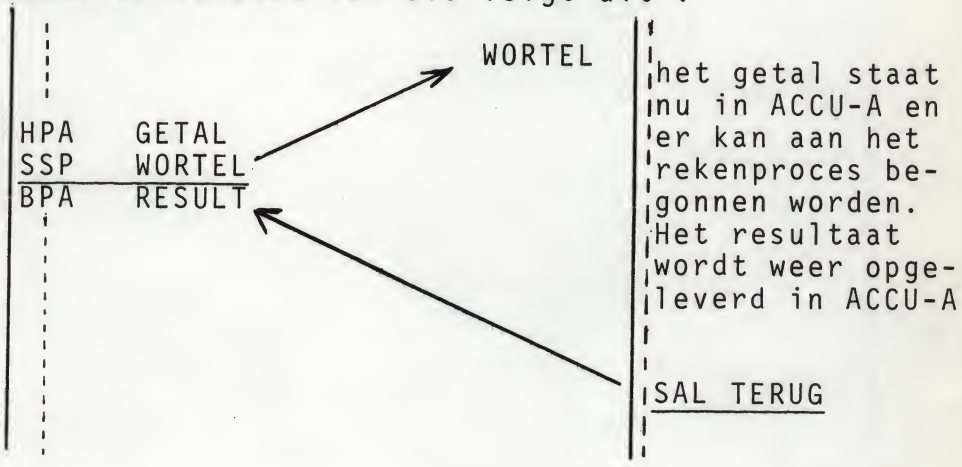
SUBPROGRAMMATECHNIEKEN

Bij het aanroepen van een subprogramma door het hoofdprogramma moeten vrijwel altijd een of meerdere gegevens meegenomen worden. Hetzelfde geldt bij terugkeer vanuit het subprogramma naar het hoofdprogramma.
Er zijn een aantal mogelijkheden om dit te bewerkstelligen.
Indien een of twee gegevens meegenomen worden verdient het gebruik van de A- en B-accumulator de voorkeur.
Bij meerdere gegevens hangt het van de omstandigheden af welke keuze men doet.

Aan de hand van voorbeelden zullen de verschillende methoden behandeld worden.

Voorbeeld 1.

Een subprogramma dat de vierkantswortel uit een getal moet berekenen heeft slechts één invoergegeven en slechts één resultaat.
Aangenomen dat het getal waaruit de wortel getrokken moet worden in de geheugenplaats met label GETAL staat en het resultaat van de worteltrekking in RESULT moet komen, dan ziet het programma er schematisch als volgt uit :



Voorbeeld 2

Met behulp van invoergegevens in ACCU-A en ACCU-B wordt be-

lasting berekend die in ACCU-A wordt opgeleverd.

LOON	GR1	GR2	gezinsstruktuur → GR3 1	2	3	4	5	6	7	8	9	10
-9423- 9480	1479	791	623	466	311	192	70	0	0	0	0	0
9481- 9538	1495	802	632	473	318	198	76	0	0	0	0	0
9539- 9595	1509	811	641	481	326	204	81	0	0	0	0	0
21885-22124	6090	3823	3480	3157	2836	2561	2287	2028	1767	1511	1257	1002
29962-30191	10068	6669	6247	5838	5434	5086	4740	4414	4091	3768	3450	3131

Figuur 1.

In de kolom "loon" staan op elke rij twee weeksalarissen in centen. Verder staat op elke rij een serie bedragen die betrekking heeft op de verplichte loonbelasting, behorende bij die salarissen.

Deze belastingbedragen zijn onderverdeeld in drie groepen te weten :

Groep 1 : ongehuwd
Groep 2 : gehuwd
Groep 3 : gehuwd met kinderen.

Bij deze laatste groep is het aantal kinderen bepalend.
De tabel wordt nu als volgt toegepast.
Het salaris van al de te onderzoeken personen moet liggen tussen twee bedragen die genoteerd staan in de kolom "loon". Natuurlijk kan zo'n salaris ook gelijk zijn aan één van die twee bedragen.
Is de rij gevonden waarbij dat optreedt, dan wordt naar de gezinsstruktuur van de betrokken persoon gekeken, waarna de verschuldigde belasting bekend is.

Ir. P.A. Tas

Gevraagd wordt nu een subprogramma te schrijven, dat de loonbelasting berekent indien de gegevens van het hoofdprogramma bestaan uit een salaris en de gezinsstruktuur.
Dit zijn n.l. de enige gegevens die noodzakelijk zijn om de loonbelasting te kunnen vinden.
We zullen voor de overzichtelijkheid een klein hoofdprogramma schrijven dat gebruik gaat maken van dit subprogramma.

We gaan uit van een aantal kaarten met op elke kaart de volgende gegevens :

kolom

10 t/m 25	een naam
30 t/m 35	een personeelsnummer
40 t/m 44	weeksalaris in centen
50 t/m 51	het getal -1 of een van de getallen 0, 1, 2 ..... 10 die de gezinsstruktuur aangeven.

In het laatste geval betekent -1 dat de betreffende persoon ongehuwd is. De andere getallen hebben betrekking op het aantal kinderen. Als beperking voor dit programma wordt aangenomen dat er niet minder zal worden verdiend dan *f*. 94,23 en niet meer dan *f*. 300,--
Verder zal het aantal kinderen niet meer dan 10 bedragen.

Gevraagd wordt een lijst met namen en loonbelasting uit te drukken op de sneldrukker resp. op de posities 10 t/m 25 en 40 t/m 44 van het papier.

Een blanko kaart fungeert als sluitkaart. Het subprogramma krijgt de naam BELDEREK.

Bij de sprong naar dit subprogramma moet het salaris in ACCU-A en de gezinsstruktuur in ACCU-B staan.

Bij terugkomst uit het subprogramma staat de belasting in ACCU-B.

```
          BGN
          BUS     81:120      buffer schoon
VERVOLG   LSK
          HAB     40:44
          KAG     5           salaris
          SOB     EIND        spring indien salaris =0
          BPB     HULP
          HAB     50:51
          KAG     2           gezinsstruktuur ⟶ ACCU-B
          HPA     HULP        salaris ⟶ ACCU-A
          SSP     BELDEREK
```

```
                BUS     30:80       bij terugkomst uit het
                KGA     5           subprogramma staat nu de
                BAB     40:44       belasting in ACCU-B
                DRU
                SAL     VERVOLG
EIND            STP
HULP            0
BELDEREK        ---
                ---                 het subprogramma BELDEREK
                ---
                ---
                SRT
```

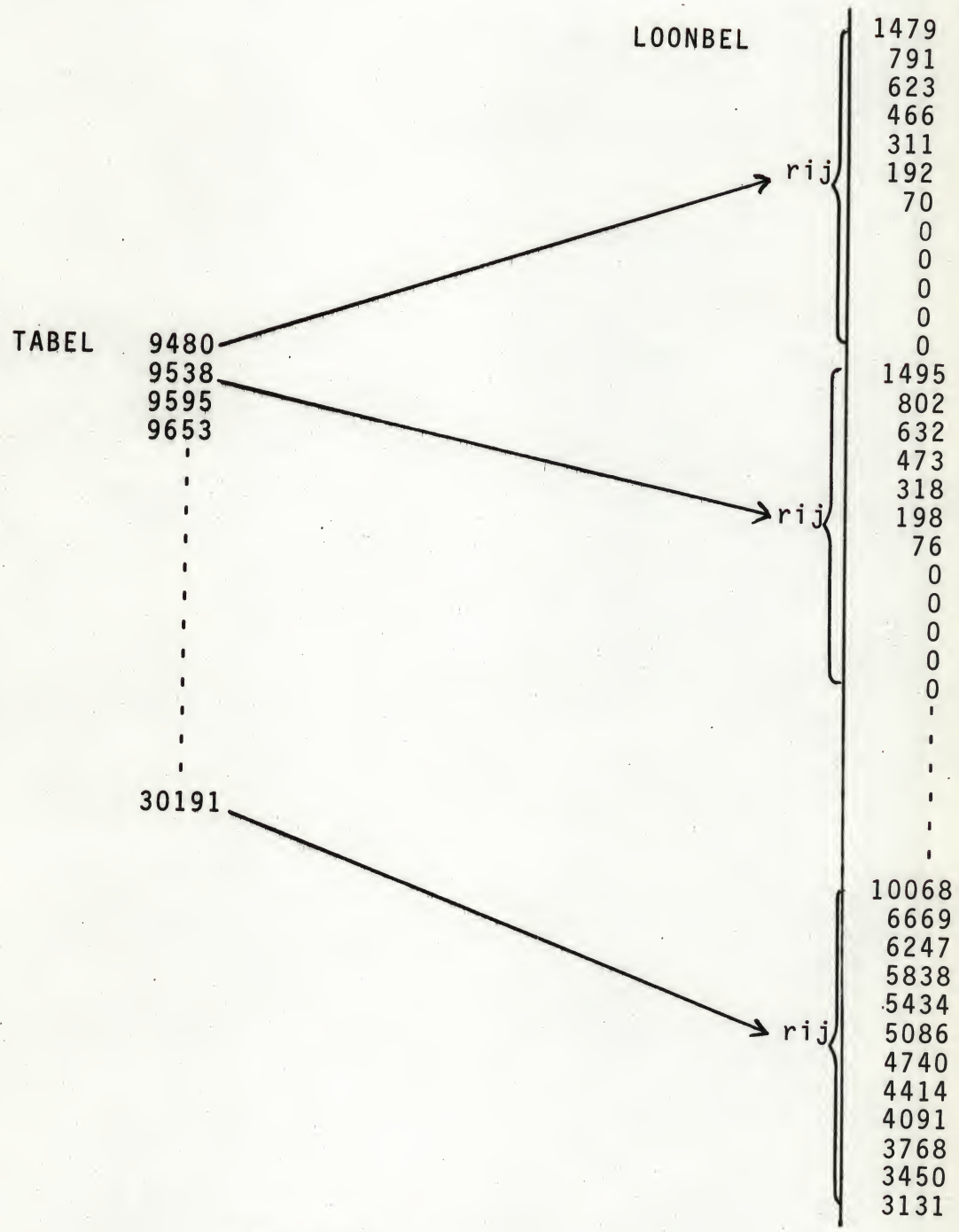
Indien een blanko kaart wordt ingelezen, wordt de buffer opgevuld met spaties.
Met de HAB 40:44 instruktie worden dan spaties naar ACCU-B gehaald.
Conversie levert dan het getal 0 op in ACCU-B.

HET SUBPROGRAMMA BELDEREK

De belastingtabel zal op de één of andere wijze in het geheugen moeten worden opgeslagen.
Hiervoor is een oplossing gekozen, waarbij gebruik wordt gemaakt van twee lijsten.
Het gedeelte van de tabel met de bedragen van de loonbelasting is in rijen achter elkaar in het geheugen opgeslagen.
Deze lijst heeft de label LOONBEL meegekregen.
Voor de lonen wordt een aparte lijst gemaakt met label TABEL, waarin achter elkaar het tweede salaris van de kolom "loon" is opgeslagen.
Het is niet nodig om beide salarissen op te slaan omdat het eerste salaris van de volgende rij 1 groter is dan het laatste salaris van de betreffende rij.
We kunnen nu zeggen dat elk getal uit de lijst TABEL betrekking heeft op 12 getallen in de lijst LOONBEL op een wijze zoals die is geschetst in figuur 2.

Ir. P.A. Tas

34



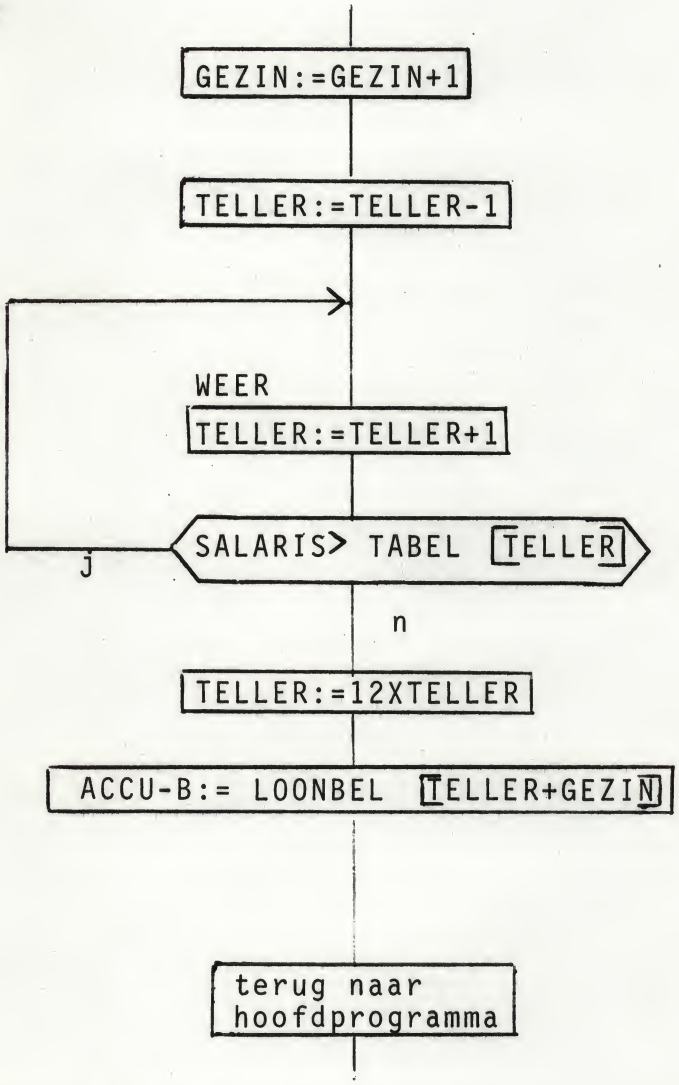
Figuur 2.

Het gegeven salaris wordt nu achtereenvolgens vergeleken met opvolgende getallen uit de lijst TABEL.

Indien:

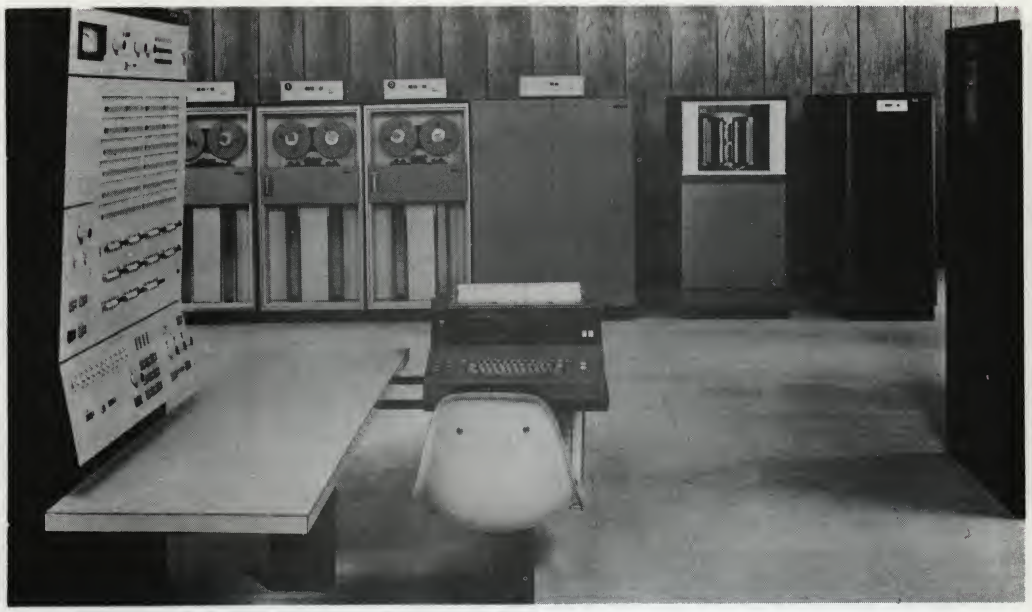
salaris $\geq$ getal. Haal het volgende uit TABEL.
salaris $\ll$ getal. De juiste loonbelasting rij is gevonden. Zoek nu met behulp van de gezinsstruktuur de goede loonbelasting.

stroomdiagram



Dit stroomdiagram behoeft enige verklaring.
De gebruikte symbolische namen, die nog niet zijn genoemd, hebben de volgende betekenis:

Optische leeseenheid als onderdeel van een computer-systeem.



Multi processing systeem (IBM-360/65)

TELLER een hulpteller
GEZIN het getal dat de gezinsstruktuur aangeeft
SALARIS het gegeven salaris

Bij binnenkomst in het subprogramma wordt GEZIN met 1 vermeerderd, waardoor het straks mogelijk wordt eenvoudig met GEZIN te modificeren. In de lus wordt het volgende getal uit TABEL vergeleken met SALARIS. U ziet hier dat TABEL wordt gemodificeerd met TELLER.
Zodra het salaris kleiner is, wordt de TELLER met 12 vermenigvuldigd, waardoor er een getal ontstaat dat kan worden gebruikt om de bijbehorende rij te vinden in de lijst LOONBEL. Door nu LOONBEL te modificeren met TELLER en GEZIN vinden we de gevraagde belasting.

Voorbeeld:

Stel het salaris is *f*. 94,95 en het aantal kinderen bedraagt 2.
Bij binnenkomst in het subprogramma : GEZIN is 2, SALARIS is 9495. Nu wordt GEZIN met 1 verhoogd dus GEZIN:=3. TELLER wordt in de lus gelijk aan 0 en dus wordt TABEL [0] vergeleken met SALARIS en dat wil zeggen:
9480 wordt vergeleken met 9495.
SALARIS is groter en dus wordt teruggesprongen in de lus.
TELLER wordt verhoogd tot 1 en nu worden 9538 en 9495 vergeleken.
Nu is echter TABEL TELLER groter.
TELLER wordt met 12 vermenigvuldigd, waardoor TELLER:=12.
Het juiste belastingbedrag is gevonden als LOONBEL wordt gemodificeerd met TELLER + GEZIN. Deze zijn samen gelijk aan 15.
LOONBEL + 15 is dus het adres waar het om gaat, en daar staat het getal 473 in.

Het subprogramma BELDEREK:

```
BELDEREK | OPB * 1        |  GEZIN:=GEZIN+1
         | BPB   GEZIN    |
         | HPB * 1        |
         | BNB   TELLER   |  -1 ⟶ TELLER
WEER     | HPB   TELLER   |
         | OPB * 1        |  TELLER:=TELLER+1
         | BPB   TELLER   |
         | MOD   TELLER   |
         | HPB   TABEL    |  TABEL [TELLER] -SALARIS <0?
         | AFB   ACCU-A   |
         | SNB   WEER     |
         |----------------|
         | HPB   TELLER   |
         | VMG * 12       |  TELLER:=12xTELLER voor de juiste rij
         | OPB   GEZIN    |     modificeren met TELLER+GEZIN
```

```
          MOD     ACCU-B
          HPB     LOONBEL
          MOD     TERUG      betreffende loonbelasting in ACCU-B
          SAL     0
TELLER    0
GEZIN     0
LOONBEL   1479
           791
           623
            .
            .
            .
            .
          3450
          3131
TABEL     9480
          9538
            .
            .
            .
            .
          30191
          VRY     BELDEREK
```

Voorbeeld 3

Het subprogramma verwacht 3 gegevens u, v en w en levert 2 resultaten k en l.
De gegevens en resultaten volgen op de SSP-opdracht.

Hoofdprogramma

```
                          SUB
SSP   SUB  ------------------>
gegeven u
gegeven v
gegeven w
resultaat k
resultaat l
volgende instruktie
          <------------------
```

Subprogramma

```
          .
          .
MOD  TERUG w ---> A
HPA  2
          .
MOD  TERUG v ---> A
HPA  1
          .
MOD  TERUG u ---> A
HPA  0
          .
MOD  TERUG k opbergen
BPA  3
          .
MOD  TERUG l opbergen
BPA  4
          .
MOD  TERUG spring terug naar het
SAL  5                juiste adres
```

Ir. P.A. Tas 38

Bij de subprogrammasprong SSP SUB wordt het adres van de instruktie volgend op de SSP SUB, opgeborgen in het register met vaste naam TERUG. Daar wordt gebruik gemaakt bij het halen van de gegevens en het opbergen van de resultaten. Uiteraard moet bekend zijn waar die gegevens in het hoofdprogramma staan en waar de resultaten naar toegebracht moeten worden.
w staat 3 plaatsen verder dan de SSP-sprong. Door nu 2 op te tellen bij (TERUG), wordt het juiste adres van w gevonden.
Dit wordt tesamen met het halen naar ACCU-A gerealiseerd in de twee opdrachten:

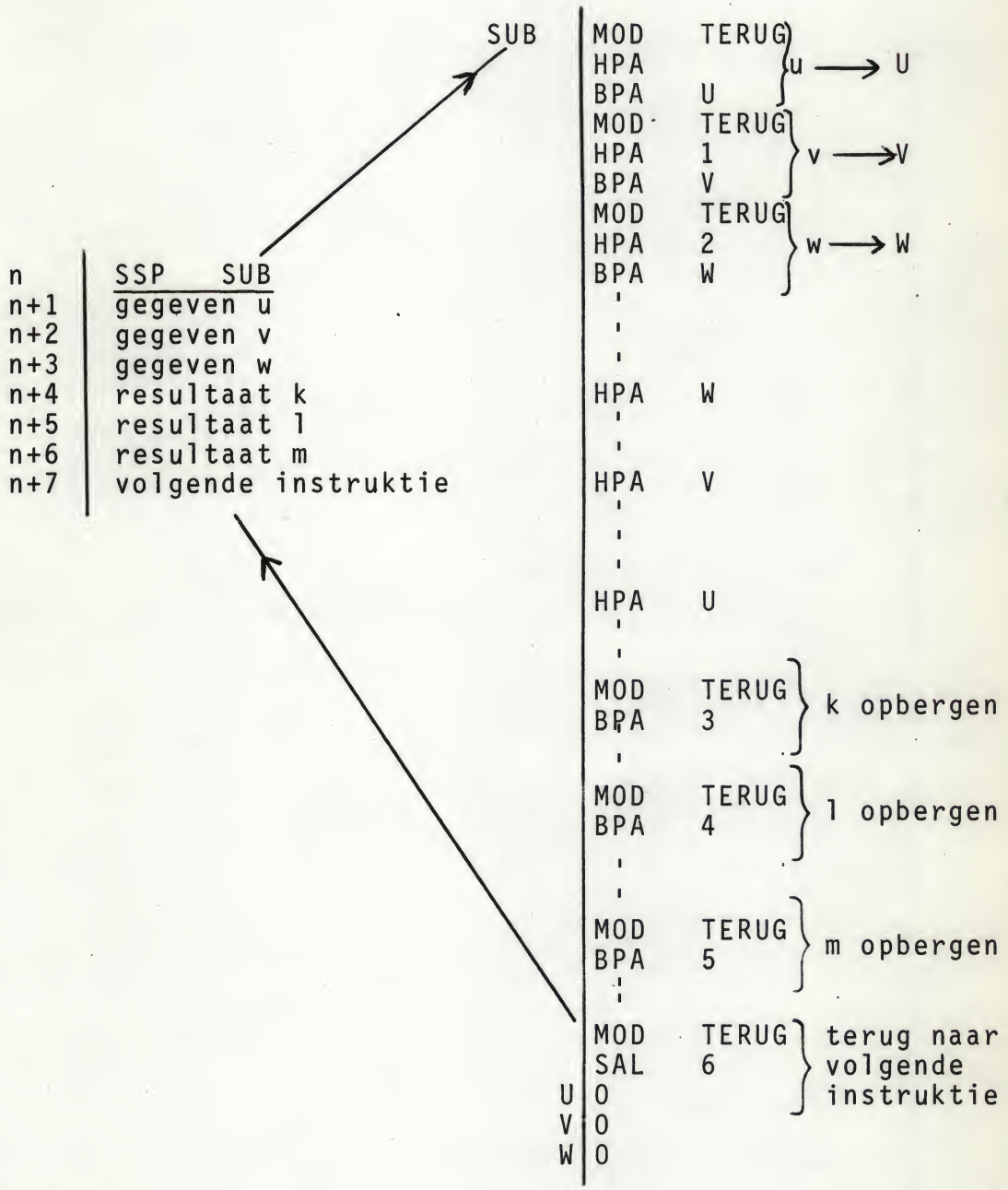
```
MOD  TERUG
HPA  2
```

Het adresgedeelte van de HPA-opdracht wordt gemodificeerd met (TERUG).

Nu wordt de opdracht uitgevoerd als HPA 2+(TERUG).
Op dezelfde wijze worden de opbergopdrachten en de sprong terug naar het hoofdprogramma verzorgd.

Moet een gegeven meer keren worden opgehaald in het subprogramma, dan moet telkens zo'n konstruktie met een modifikatie-opdracht worden geschreven.
Dat is vervelend en bovendien niet erg overzichtelijk.
Het verdient dan ook voorkeur de gegevens eerst in het subprogramma op te halen en zo lang op te bergen in daarvoor gereserveerde woorden.
Dan kan men verder in het subprogramma verwijzen naar de inhoud van die gereserveerde woorden.
Het schema kan er dan als volgt uit komen te zien :

Ir. P.A. Tas

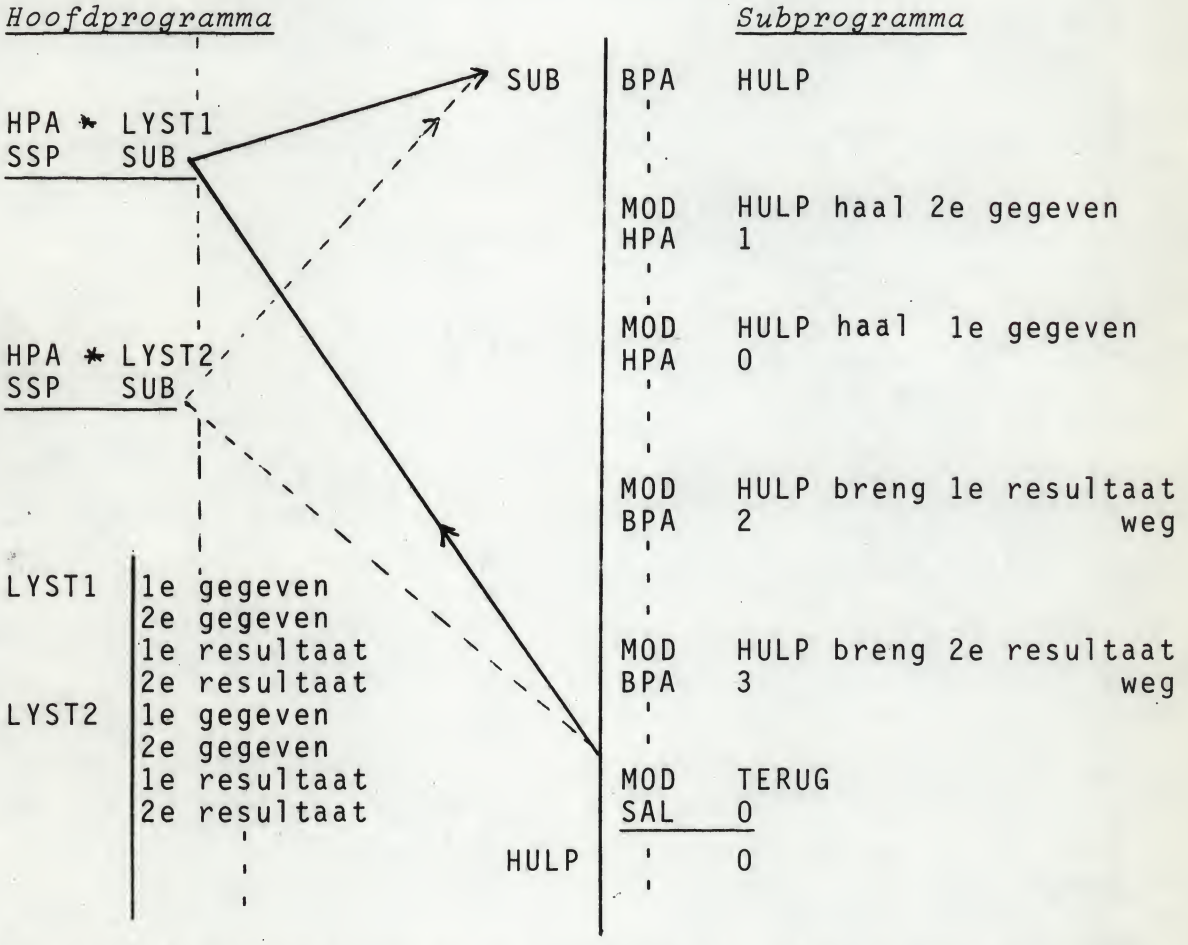


Voorbeeld 4.

Gegevens en resultaten zijn vanaf een willekeurige plaats op elkaar volgend in het hoofdprogramma opgeborgen. Het adres van deze plaats wordt naar ACCU-A gehaald, voordat het subprogramma aangesprongen wordt. Het hoofdprogramma springt naar het

Ir. P.A. Tas

subprogramma tweemaal met verschillende gegevens.



De ster-faciliteit zorgt er voor dat in plaats van de inhoud van een geheugenplaatsadres het adres zelf als operand gebruikt wordt. Het invoerprogramma verzorgt het echte adres behorende bij LYST1.
HPA * LYST1 brengt het adres dat overeenkomt met LYST1 naar de accumulator. Dit gegeven wordt als invoer voor het subprogramma gebruikt en door het subprogramma opgeborgen in een hulpregister :
HULP.
Het eerste gegeven wordt bijv. opgehaald door :

```
MOD  HULP                    (HULP) = LYST1
HPA  0
```

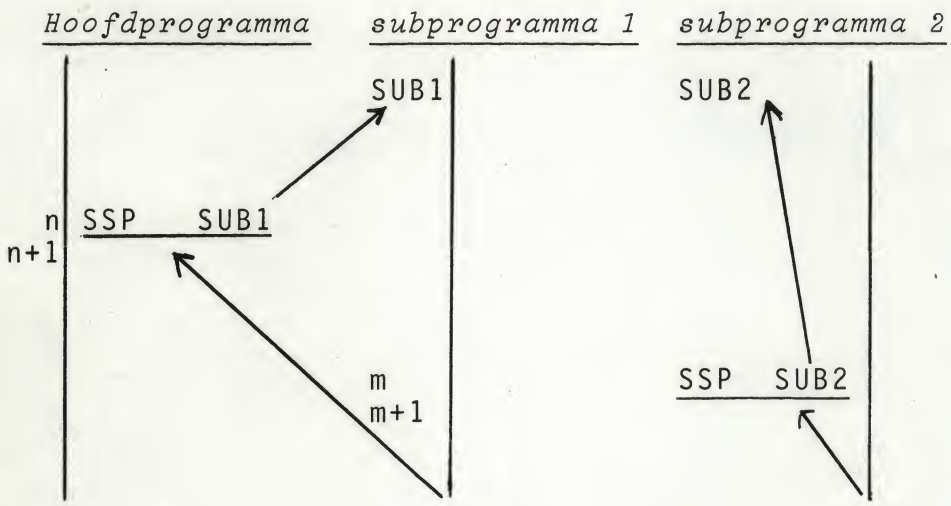
Dit is identiek met : HPA 0+ (HULP)=HPA 0+LYST1 = HPA LYST1.

Voorbeeld 5.

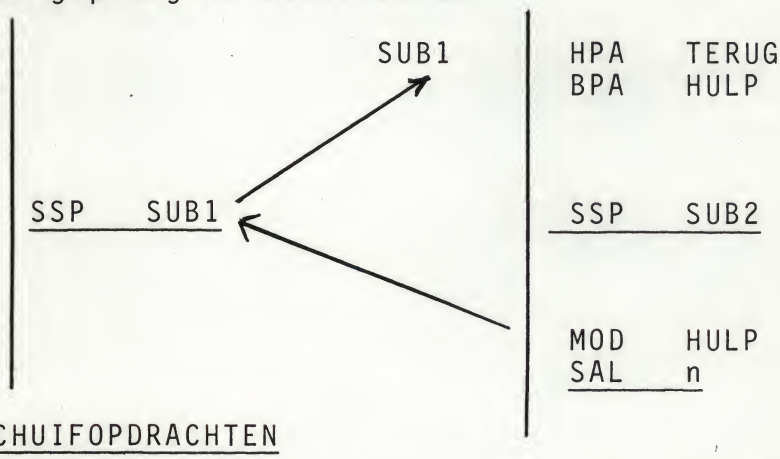
Het komt ook wel voor dat in een subroutine een andere subroutine

Ir. P.A. Tas

wordt aangeroepen. Dit wordt schematisch weergegeven in onderstaande figuur.



Bij de subprogrammasprong vanuit het hoofdprogramma naar het subprogramma1 wordt het terugkeeradres n+1 opgeborgen in TERUG. Daarna wordt subprogramma1 doorlopen, totdat de SSP sprong naar subprogramma2 wordt ontmoet.
Die zorgt er voor dat er naar SUB2 wordt gesprongen en het bijbehorende terugkeeradres m+1 wordt in TERUG gebracht, waardoor het oude terugkeeradres naar het hoofdprogramma wordt vernietigd.
Dat is ontoelaatbaar, omdat dan de mogelijkheid om van subprogramma 1 naar het hoofdprogramma terug te keren, verloren is gegaan.
Daarom moet het terugkeeradres voor het hoofdprogramma in subprogramma 1 veilig worden gesteld.
Dit kan zeer eenvoudig geschieden, door de inhoud van TERUG op te bergen in een hulpgeheugenwoord en later met dit hulpwoord de terugsprong te modificeren.



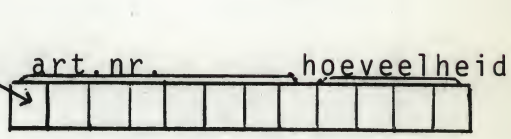
DECIMALE SCHUIFOPDRACHTEN

Om geheugenruimte te besparen kan het nuttig zijn om ver-

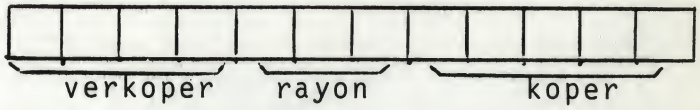
schillende soorten informatie in een geheugenwoord samen te voegen.

Voorbeelden van samenvoeging.

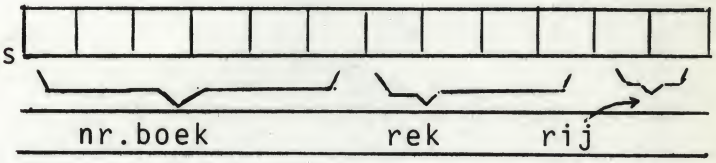
artikelnummer met aantal



nr. verkoper, rayonnr.
nr. koper



nr. van een boek, plaats waar het staat



Voor het *"pakken"* en *"uitpakken"* van meerdere numerieke gegevens in een woord kan men onder andere gebruik maken van de decimale schuifopdrachten.
Verder is het mogelijk door schuiven te vermenigvuldigen met machten van 10.

De schuifopdracht werkt tegelijkertijd op ACCU-A en ACCU-B. De gehele inhoud van de accumulatoren wordt een aantal tetraden naar links of naar rechts "geschoven".
Wordt er naar links geschoven, dan zullen de meest linkse tetraden van ACCU-A verloren gaan, terwijl de meest rechtse tetraden van ACCU-B opgevuld worden met nullen.
Naar rechts schuiven betekent dat de meest rechtse tetraden van ACCU-B verloren gaan, terwijl de meest linkse tetraden van ACCU-A met nullen gevuld zullen worden.
Tekentetraden schuiven niet mee.

(A) en (B) vóór de schuifopdracht.

+ 1 2 3 4 5 6 7 8 9 9 9	+ 0 1 2 3 1 1 2 1 1 0 1

na de opdracht : schuif 3 decimale plaatsen naar links

+ 4 5 6 7 8 9 9 9 0 1 2	+ 3 1 1 2 1 1 0 1 0 0 0

(A) en (B) vóór de schuifopdracht.

+ 1 3 3 1 3 3 1 2 3 4 5	+ 0 0 0 0 0 0 0 0 0 0 0

na de opdracht : schuif 2 decimale plaatsen naar rechts

+ 0 0 1 3 3 1 3 3 1 2 3	+ 4 5 0 0 0 0 0 0 0 0 0

De opdrachten :

ABL n Schuif n decimale plaatsen naar links in AB. Rechts in ACCU-B worden nullen (behorende bij het teken) bijgeschoven.

ABR n Schuif n decimale plaatsen naar rechts in AB. Links in ACCU-A worden nullen (behorende bij het teken) bijgeschoven.

Er is een belangrijk nevenverschijnsel bij deze opdrachten. De inhouden van A en B worden beschouwd als zijnde een dubbele-lengte-getal. Voordat de schuifopdracht plaatsvindt worden de tekens van A en B gelijk gemaakt.
Indien (A) = + 0 dan wordt het teken van A gelijk gemaakt aan het teken van B.
Indien (A) ≠ 0 dan wordt het teken van B gelijk gemaakt aan die van A, waarbij een eventuele overdracht in A wordt verwerkt.

Voorbeeld

A	B
+ 0 0 0 0 0 1 2 3 4 5 6	- 5 3 1 0 0 0 0 0 0 0 0

Nu de opdracht ABL 2.
Eerst worden de tekens van de registers gelijk gemaakt. Hierbij wordt geleend van A om B positief te maken.

+ 0 0 0 0 0 1 2 3 4 5 5	+ 4 6 9 0 0 0 0 0 0 0 0

Daarna

+ 0 0 0 1 2 3 4 5 5 4 6	+ 9 0 0 0 0 0 0 0 0 0 0

n=0 is toegestaan bij de schuifopdrachten en dit is dan ook de beste manier om de inhouden van de registers A en B hetzelfde teken te geven.

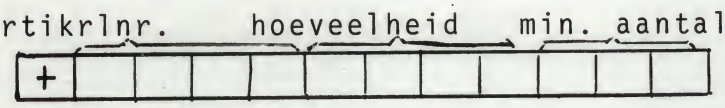
Voorbeeld

Vroeger is behandeld een voorbeeld van een voorraadadministratie met behulp van 2 lijsten.
In plaats van 2 lijsten wordt nu gebruik gemaakt van 1 lijst.
De benaming van het art. komt te vervallen.
Alle gegevens van een art. en van de voorraad worden in één woord opgeborgen.
Behalve het artikelnummer en de hoeveelheid in voorraad be-

vat het woord ook nog een"minimum aantal".
Zodra de voorraad onder het minimum aantal daalt wordt dit door het programma gesignaleerd.

Voor de eenvoud zal de uitgang "artikel niet aanwezig" zoals in vorig voorbeeld achterwege gelaten worden.

Een woord uit de lijst heeft nu de volgende gedaante:



Voorbeeld van een lijst.

```
   art.nr. hoeveelheid      min. aantal
LYST   0033 0141 020
       0035 0210 010
       0045 1003 100
       0046 0028 003
       0047 0011 002
       0048 0009 020
```

De artikelen staan in volgorde in de lijst.
De mutatiekaarten zijn ook op artikelnummer gesorteerd.
Het is dus niet meer noodzakelijk om voor elke mutatiekaart de gehele lijst van begin af aan te doorzoeken, er kan nu verder gezocht worden vanaf de plaats waar de vorige kaart in de lijst gebleven is.

De kaart bevat:

in kolom	
4 - 7	artikelnummer
10 - 14	aantal met teken dat de voorraad verandert.

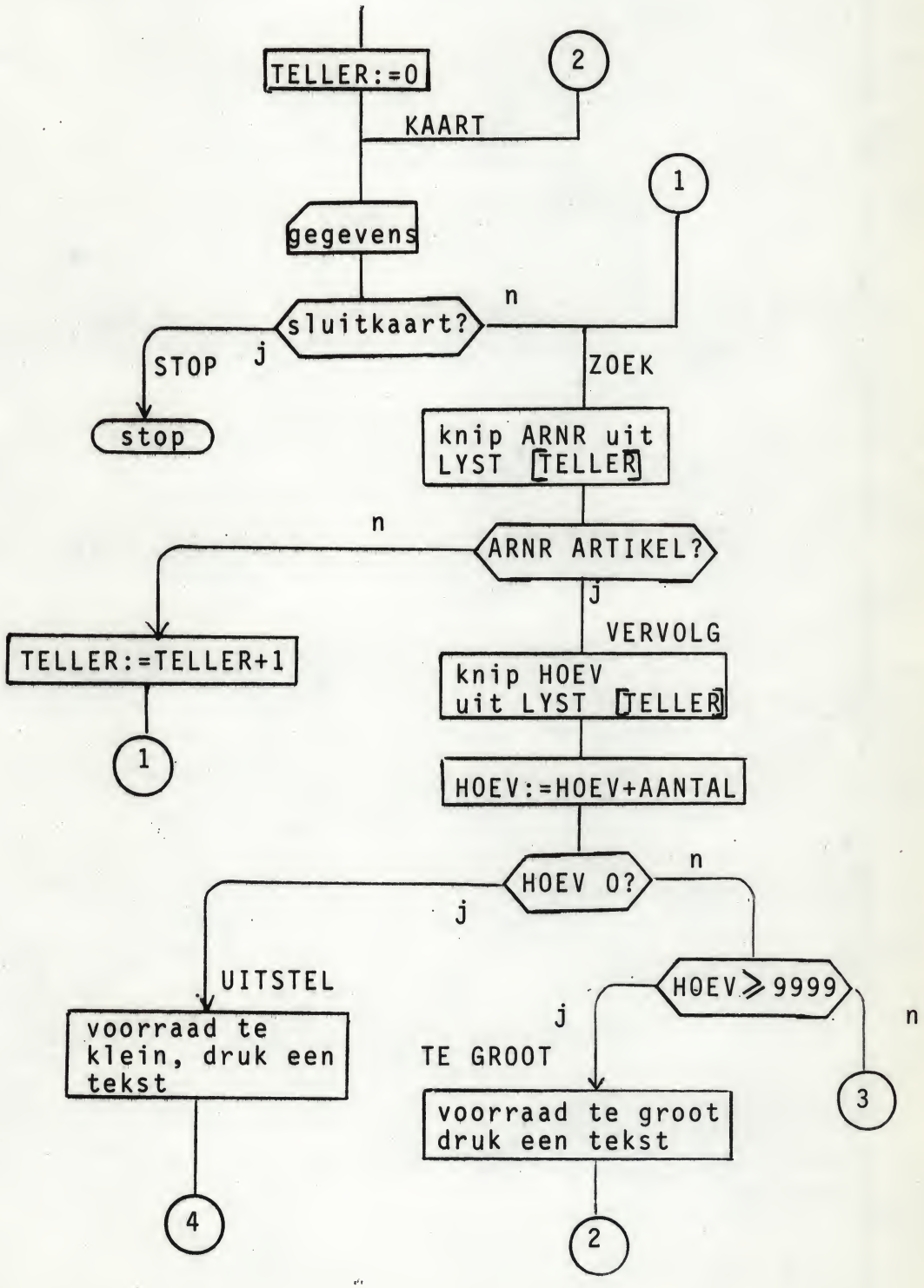
Het kan zijn dat de voorraad niet toereikend is voor de bestelling: in dat geval moet de bestelling uitgesteld en de voorraad bijgevuld worden.
Is na de afschrijving het aantal $<$minimum aantal, dan voorraad bijvullen.

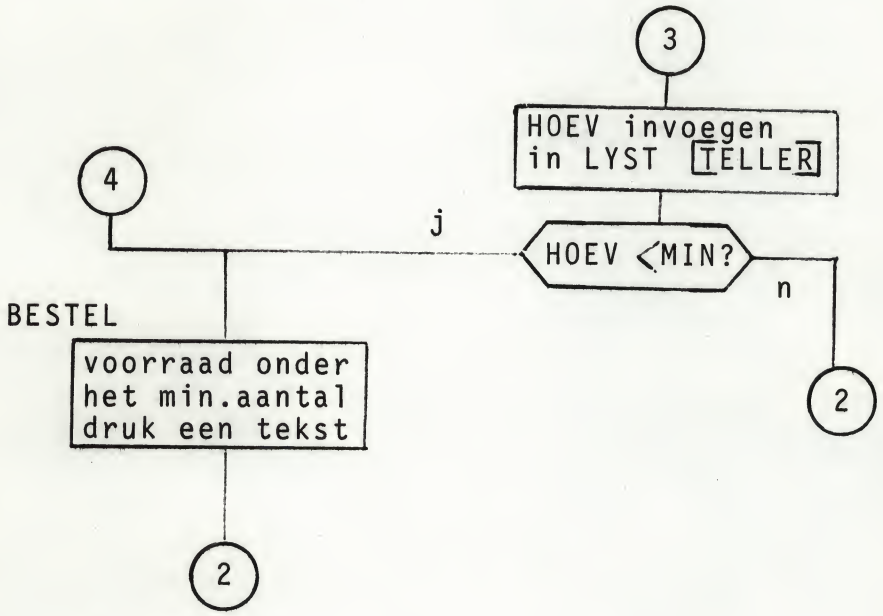
Er wordt getest op een sluitkaart met in kolom 4 t/m 7 het woord "EIND".

Gebruikte namen in het stroomdiagram:

ARTIKEL	artikelnummer in de kaart
AANTAL	voorraadverandering in de kaart

ARNR artikelnummer van een woord uit de lijst
HOEV voorraad van een bepaald artikelnummer uit de lijst
MIN minimum aantal van een artikelnummer uit de lijst

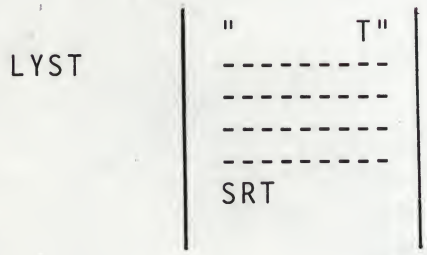





```
                BGN
STOP            STP     STOP+1
                BUS     80:120
                BSA     TELLER      TELLER:=0
                BPA     TELLER
KAART           LSK
                HAB     4:7
                HPA     TEKST       Is het een sluitkaart?
                SAB     STOP
                KAG     4
                BPB     ARTIKEL     artikelnummer ---> ARTIKEL
                HAB     10:14
                KAG     5           verandering in de voorraad --> AANTAL
                BPA     AANTAL
                HPA  *              0 ---> ACCU-A
ZOEK            MOD     TELLER
                HPB     LYST        art.nr. uit de lijst komt in ACCU-A
                ABL     4
                HPB     ARTIKEL     vergelijk art. nrs.
                SAB     VERVOLG
                HPA     TELLER
                OPA  *  1           TELLER:=TELLER+1
                BSA     TELLER      0 ---> ACCU-A
                SAL     ZOEK
VERVOLG         MOD     TELLER
                HPA     LYST
                ABL     4           hoeveelheid uit het woord geknipt
                ABR     7
                OPA     AANTAL      verandering voorraad
```

```
          SNA      UITSTEL
          BPA      HOEV          wegbergen
          AFA   *  9999
          SPA      TEGROOT       hoeveelheid ≥ 9999
          MOD      TELLER
          HPA      LYST
          ABL      8             min. aantal ⟶ MIN
          ABR      8
          BPA      MIN
          MOD      TELLER
          HPB      LYST
          ABR      7             voeg in het woord de nieuwe voor-
          ABL      4             raad en het min. aantal en breng
          OPB      HOEV          naar LYST  [TELLER]
          ABL      3
          OPB      MIN
          MOD      TELLER
          BPB      LYST
          HPA      MIN
          AFA      HOEV
          SPA      BESTEL
          ----------
          SAL      KAART
UITSTEL   HPA      TEKST+1
          HPB      TEKST+2
          BAB      20:35
          HPB      TEKST+3
          BAB      36:36
BESTEL    HPB      TEKST+4
          BAB      40:45
          DRU
          SAL      KAART
TEGROOT   HPA      TEKST+1
          HPB      TEKST+5
          BAB      20:35
          HPB      TEKST+6
          BAB      36:36
          DRU
          SAL      KAART
HOEV      0
MIN       0
ARTIKEL   0
AANTAL    0
TELLER    0
TEKST     "OOOOEIND"
          "VOORRAAD"
          " TE KLEI"
          "        N"
          "BESTEL  "
          " TE GROO"
```

Ir. P.A. Tas



Opgaven les 4 008

A.1

SERIE-A

Naam

Adres

Woonplaats

Kursistennr.

Voor de onderstaande vragen dient U het rondje op te vullen van de antwoorden die volgens U goed zijn. Wij wijzen U er echter bij voorbaat op dat bij een vraag meerdere antwoorden en zelfs alle antwoorden goed kunnen zijn. Het tegenovergestelde kan zich ook voordoen.M.a.w. er kan op een bepaalde vraag ook geen enkel antwoord goed zijn.

Na het oplossen van de vragen dient U de pagina's van Serie-A in te sturen aan E.C.S. Postbus 2 Heerlen.

Veel succes.

Vraag 1.

O Het adresgedeelte van een instruktie kan tijdens de uitvoering van het programma nooit worden veranderd.

O Het adresgedeelte van een instruktie kan tijdelijk worden veranderd, indien de instruktie zich in het besturingsorgaan bevindt.

O Door de MOD instruktie kan het adresgedeelte van een instruktie in het geheugen voor onbepaalde tijd worden veranderd.

Vraag 2.

O De SERA computer kan aan een woord in het geheugen duidelijk zien of de inhoud een instruktie dan wel een getal is.

O De SERA computer kan aan een woord in het geheugen niet zien of de inhoud een instruktie dan wel een getal is.

O Het is mogelijk om met een instruktie normaal te rekenen en een getal in een woord als een instruktie op te vatten.

Vraag 3.

O Tijdens het verwerken van een MOD instruktie wordt er in het geheugen niets veranderd.

O Een MOD instruktie veroorzaakt dat het adres van de MOD instruktie in het modificatieregister wordt geladen.

O Een MOD instruktie veroorzaakt het laden van het MOD-instruktieadresgedeelte in het modificatieregister.

O Door een MOD instruktie wordt de inhoud van het MOD-instruktie-adres in het modificatieregister geladen.

Opgaven les 4

SERIE-A

Naam

Woonplaats

Vraag 4.
- O Een instruktie wordt alleen gemodificeerd met het modificatieregister, indien de inhoud van het modificatieregister ongelijk is aan nul.
- O Van de instruktie volgend op een MOD instruktie wordt alleen het adresgedeelte gemodificeerd.
- O De instruktie volgend op een MOD instruktie wordt steeds met het volledige modificatieregister gemodificeerd.

Vraag 5.

Gegeven :

(ACCU A) = 100
(1000) = 200
(2000) = 300
(2200) = 400
(3000) = 500

Na uitvoering van de volgende 2 instrukties :

```
    :
MOD   1000
OPA   2000
```

is de inhoud van ACCU A :

- O 400
- O 500
- O 600

Vraag 6.
- O De SERA machine bezit geen indexgeheugen.
- O Door de aanwezigheid van een indexgeheugen wordt de programmering meer flexibel.
- O Alle index-registeropdrachten zijn gemodificeerde opdrachten.

Vraag 7.

Gegeven :

(ACCU B) = 500
(100) = 50
(200) = 400
(250) = 200
(300) = 400
(1000) = 100
(1200) = 350
(1300) = 400
(1400) = 200
(1500) = 500

SERIE-A

Naam

Woonplaats

Na uitvoering van de volgende 3 instrukties :

```
        ¦
        ¦
    MOD   100
    MOD   200
    MOD  1000
```

is de inhoud van ACCU-B :

0 100

0 300

0 200

0 400

0 0

Vraag 8.

0 De index tussen vierkante haakjes is in de SERA-code slechts als een schrijfwijze ingevoerd.

0 De index tussen vierkante haakjes kan men in de SERA instrukties rechtstreeks terug vinden.

0 De index mag ook een rekenkundige uitdrukking zijn.

Vraag 9.

Het doorgeven van gegevens en resultaten van hoofdprogramma naar subprogramma en omgekeerd kan op de volgende wijze plaatsvinden :

0 Via een accu.

0 Via beide accu's.

0 Door vlak achter de subprogrammasprong ruimte te reserveren voor de gegevens en de resultaten.

0 Door het adres van de gegevens- en resultatenvelden die zich ergens willekeurig in het hoofdprogramma achter elkaar bevinden in een der accu's te laden.

0 Dit gebeurt maar zeer zelden omdat de programmeurs onderling precies weten op welke adressen zich de gegevens bevinden en waar de resultaten naar toe moeten.

Vraag 10.

0 Een subprogramma kan nooit een ander subprogramma aanroepen.

0 Voor het aanroepen van een subprogramma door een ander subprogramma, wordt geen gebruik gemaakt van het TERUG register.

0 Indien een subprogramma een ander subprogramma aanroept, dan dient van te voren de inhoud van het TERUG register gered te worden.

Opgaven les 4

008

A.4

SERIE-A

Naam

Woonplaats

Vraag 11. Decimale schuifopdrachten hebben de volgende toepassingsmogelijkheden :

- O Vermenigvuldigen met machten van 10.
- O Delen door machten van 2.
- O Pakken van meerdere gegevens in een woord.
- O Accu's schoonmaken.
- O Tekens van de accu's A en B gelijk maken.
- O Zetten van het alarmregister.
- O Afronden van getallen.

Vraag 12.

- O Decimale schuifopdrachten hebben betrekking op de accu's A en B als een 96-bitwoord.
- O Decimale schuifopdrachten hebben betrekking op of ACCU-A of ACCU-B.
- O Bij decimale schuifopdrachten worden de tekens van A en B meegeschoven.
- O Bij decimale schuifopdrachten worden de tekens van A en B niet meegeschoven.

Vraag 13.

- O Bij decimale schuifopdrachten gaat er nooit informatie verloren (m.a.w. het schuiven is rondgekoppeld).
- O Bij decimaal schuiven naar links worden er rechts in ACCU-B spatietetraden ingeschoven.
- O Bij decimaal schuiven naar rechts worden er links in ACCU-A nullen ingeschoven.

Opgaven les 4

SERIE-B

Onderstaande vragen en opgaven dient U allen voor Uzelf uit te werken. De opgaven voorzien van drie kruisjes (+++) dient U echter uit te werken op het bijgevoegde uitwerkpapier en in te sturen ter beoordeling aan E.C.S. Postbus 2 Heerlen.

Vraag 1.a. Geef voor de 2 manieren, waarop men een instruktie tijdens het uitvoeren van een programma kan wijzigen een voorbeeld in de vorm van een klein programmadeeltje.

b. Hoe noemt men deze methodes en verklaar met enkele woorden het verschil tussen het effect voor de verdere uitvoering van het programma in beide gevallen?

c. Waarom zou men het wijzigen van een instruktie ook wel "adresrekening" kunnen noemen?

Vraag 2.a. Geef het schema van het SERA-besturingsorgaan

+++ b. Schets kort de gang van zaken binnen de verschillende registers voor de uitvoering van de volgende instrukties :

- MOD
- SSP
- SAL

Vraag 3. Gevraagd het stroomschema en 2 SERA-programma's (een keer met "voor de voet schrijven", een keer met behulp van de MOD opdracht) voor het volgende probleem:

+++

Gegeven : een onbekend aantal ponskaarten; tussen deze kaarten bevinden zich er 100 met een ponsing voor de hexade C in een kolom die deelbaar is door 10 (dus een van de kolommen 10 , 20 ... 80)

Gevraagd: pons deze 100 kaarten weer uit met de waarde van het kolomnummer gedeeld door 10 op de plaats van de C.

Opmerking : Bedenk dat de bufferopdrachten 2 adressen bevatten, die beide aangepast (gemodificeerd) moeten worden.

Vraag 4. In ACCU-B bevindt zich een adres D; dus (B) = D

Gevraagd : breng (D) naar ACCU-A m.b.v. een of

SERIE-B

meer modificatie-opdrachten en de instruktie HPA.

Vraag 5. Wat is het effect van :

+++ a.
```
MOD   TELLER
HPA   LYST
```

b.
```
MOD * TELLER
HPA   LYST
```

Vraag 6.a. Verklaar aan de hand van een voorbeeld de werking van een z.g. "laddersprong"

b. Voor welke handelingen wordt deze toegepast? (m.a.w. welke andere instrukties worden hierdoor vervangen)

Vraag 7. Maak zowel een stroomdiagram als een programma in SERA-code voor het volgende probleem.
In te lezen zijn een onbekend aantal groepjes kaarten, die ieder uit een verschillend aantal kaarten kunnen bestaan. De eerste kaart van iedere groep heeft in de kolommen 1 en 2 de volgende ponsing ** .Het inlezen van de kaarten is ten einde, indien de sluitkaart (kenmerk : ZZ in klom 1 en 2) of een beginkaart zonder sterretjes in de kolommen 1 en 2 wordt ingelezen. Iedere beginkaart geeft in de kolommen 3 en 4 het totaal aantal kaarten voor die groep aan (incl. de beginkaart). Elke volgkaart in een groep bevat in de kolommen 10 t/m 15 een getal.

Gevraagd :

a. per groep : - aantal volgkaarten in die groep
- som van de getallen uit de volgkaarten van die groep

b. na de laatste kaart : - aantal groepen
- som van alle groeptotalen.

Vraag 8. +++ Schrijft U eens op 2 verschillende manieren (afhankelijk van het doorgeven van gegevens en resultaten tussen hoofdprogramma en subprogramma) een subprogramma voor de volgende berekening.

$$x = \frac{(a+b)\ c}{d}$$

Het resultaat x wordt steeds in ACCU-B teruggegeven.A, b, c, en d zijn de parameters, die door het hoofdprogramma aan het subprogramma worden gepresenteerd.

Opgaven les 4

SERIE-B

Vraag 9.a. Formuleer de decimale schuifopdrachten.

b. Wat is de kleinste hoeveelheid bits die met een decimale schuifopdracht kan worden verschoven?

c. Hoe gedragen zich de tekenbits bij het decimaal schuiven?

d. Bij het decimaal schuiven gaat informatie verloren en er komt bij. In welke vorm?

Vraag 10. Wat is de betekenis van de volgende instrukties:
+++

a. ABL 0

b. ABR 11

c. ABL 22

Vraag 11. Voert U de instruktie ABL 5 eens voor de volgende accu-inhouden uit.
Geef als resultaat de volledige inhoud (in tetrades) van ACCU-A en ACCU-B na uitvoering van de instruktie.

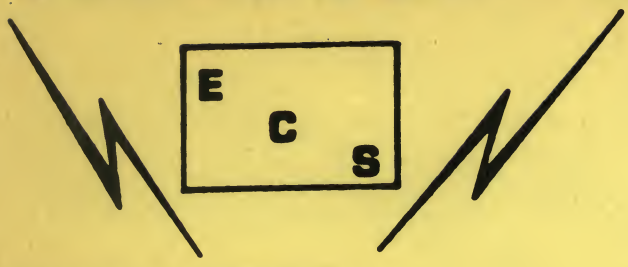
a. A = +65213 B = -2142651

b. A = -1234 B = +56789

c. A = +0 B =-9876543210

Vraag 12. In ACCU-B staat tetraal een bedrag in centen. Schrijf een programmadeel waardoor dit zelfde bedrag in guldens (met komma, zonder teken en zonder significante nullen) in de buffer komt te staan op de posities 10 t/m 22, rechts aangesloten.

european computer school

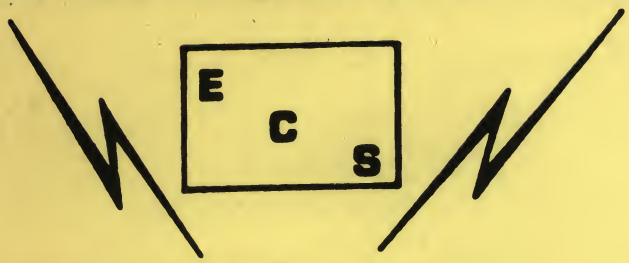


PASSAGE 4-6. HEERLEN

Naam :
Adres :
Woonplaats :
Kursistennr. :

U I T W E R K P A P I E R

european computer school

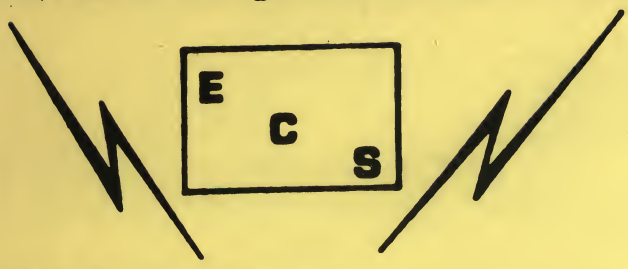


PASSAGE 4-6. HEERLEN

Naam :
Adres :
Woonplaats :
Kursistennr. :

U I T W E R K P A P I E R

european computer school



PASSAGE 4-6. HEERLEN

Naam :
Adres :
Woonplaats :
Kursistennr. :

U I T W E R K P A P I E R